



**DataLab**  
*Version 0.18.2*

**mars 17, 2025**



---

## Table des matières

---

<b>1</b>	<b>Premiers pas</b>	<b>3</b>
1.1	Installation . . . . .	4
1.2	Cas d’usage, principales fonctionnalités et points forts . . . . .	9
1.3	Fonctionnalités principales . . . . .	11
1.4	Tutoriels . . . . .	14
<b>2</b>	<b>Fonctionnalités</b>	<b>81</b>
2.1	Validation . . . . .	81
2.2	Fonctionnalités générales . . . . .	90
2.3	Traitement du signal . . . . .	133
2.4	Traitement d’image . . . . .	159
<b>3</b>	<b>API</b>	<b>195</b>
3.1	Algorithmes (cdl.algorithms) . . . . .	195
3.2	Paramètres (cdl.param) . . . . .	209
3.3	Modèle de données (cdl.obj) . . . . .	248
3.4	Computation (cdl.computation) . . . . .	282
3.5	Objets proxy (cdl.proxy) . . . . .	349
3.6	GUI . . . . .	366
3.7	Fenêtre principale . . . . .	367
3.8	Panneau . . . . .	373
3.9	Gestionnaire d’actions . . . . .	389
3.10	Vue des objets . . . . .	393
3.11	Gestionnaire de visualisation . . . . .	396
3.12	Editeur de ROI . . . . .	399
3.13	Processeur . . . . .	400
3.14	Docks . . . . .	415
3.15	E/S HDF5 . . . . .	416
<b>4</b>	<b>Contribuer</b>	<b>417</b>
4.1	Partagez vos idées et vos expériences . . . . .	417
4.2	Partagez vos connaissances scientifiques/techniques . . . . .	417
4.3	Contribuer aux nouvelles fonctionnalités . . . . .	418
4.4	Développer de nouvelles fonctionnalités . . . . .	418
	<b>Index des modules Python</b>	<b>451</b>



DataLab est une **plateforme open-source de traitement et de visualisation de signaux et d'images** pour la recherche, l'éducation et l'industrie. S'appuyant sur la richesse de l'écosystème scientifique Python<sup>1</sup>, DataLab est le complément idéal de vos flux de travail d'analyse de données car extensible avec votre code Python via *Plugins* ou directement depuis *vos IDE* ou *vos notebooks Jupyter*. Rendez-vous sur *Installation* pour démarrer !

Pour voir DataLab en action immédiatement, vous avez deux options :

- Lisez ou regardez nos *Tutoriels*,
- Essayez DataLab en ligne, sans installation, en utilisant notre *environnement Binder*.

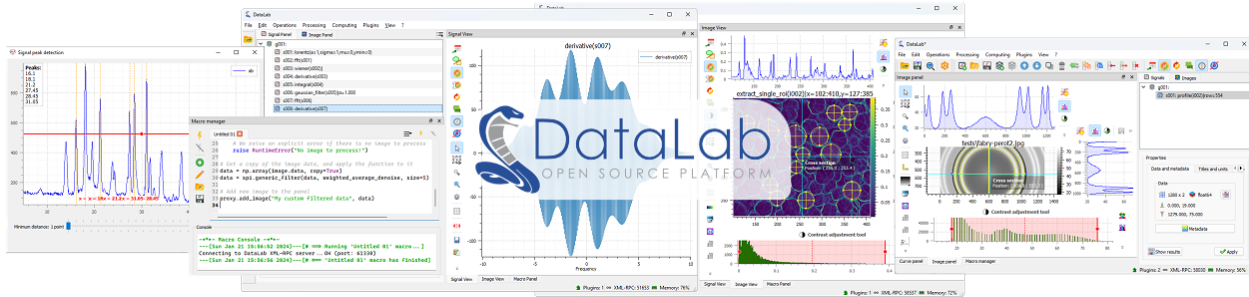


FIG. 1 – Visualisation de signaux et d'images dans DataLab



FIG. 2 – DataLab est propulsé par PlotPyStack, les outils de visualisation et d'interface graphique scientifiques Python-Qt.

1. Les fonctionnalités de traitement de DataLab sont principalement basées sur les bibliothèques NumPy, SciPy, scikit-image, OpenCV et PyWavelets. Les capacités de visualisation de DataLab sont basées sur la boîte à outils PlotPyStack, un ensemble de bibliothèques Python pour la réalisation d'applications scientifiques avec des interfaces graphiques Qt.



# CHAPITRE 1

---

## Premiers pas

---

DataLab est une plateforme ouverte de traitement de signaux et d'images, conçue pour être utilisée par les scientifiques, ingénieurs et chercheurs du monde académique et industriel, tout en offrant la fiabilité d'un logiciel industriel. C'est un logiciel polyvalent qui peut être utilisé pour un large éventail d'applications, de l'analyse de données simple aux tâches complexes de traitement de signaux et d'analyse d'images.

DataLab s'intègre parfaitement dans votre flux de travail grâce à trois modes de fonctionnement principaux :

---

**Application autonome**, avec une interface graphique qui vous permet d'interagir avec vos données et de visualiser les résultats de votre analyse en temps réel.

**Bibliothèque Python**, vous permettant d'intégrer les fonctions de DataLab (ou les interfaces graphiques) dans vos propres scripts et programmes Python ou des notebooks Jupyter.

**Piloté à distance** depuis votre propre logiciel, ou depuis un IDE (par exemple Spyder) ou un notebook Jupyter, en utilisant l'API de DataLab.

---

DataLab s'appuie sur la puissance de Python et de son écosystème scientifique, grâce à l'utilisation des bibliothèques suivantes :

- **NumPy** pour le calcul numérique (tableaux, algèbre linéaire, etc.)
- **SciPy** pour le calcul scientifique (interpolation, fonctions spéciales, etc.)
- **scikit-image** et **OpenCV** pour le traitement d'images
- **PyWavelets** pour la transformée en ondelettes
- **PlotPyStack** pour la visualisation interactive de données basée sur Qt

## 1.1 Installation

Cette section fournit des informations sur l'installation de DataLab sur votre système. Une fois installé, vous pouvez démarrer DataLab en exécutant la commande `cdl` dans un terminal, ou en cliquant sur le raccourci DataLab dans le menu Démarrer (sous Windows).

### Voir aussi :

Pour plus de détails sur l'exécution de DataLab et ses options en ligne de commande, voir *Ligne de commande*.

### 1.1.1 Modes d'installation

DataLab est disponible sous plusieurs formes :

- En tant que *Paquet Conda*.
- Un paquet Python qui peut être installé à l'aide du *Gestionnaire de paquets pip*.
- Windows En tant qu'application autonome, qui ne nécessite pas d'installation de Python. Il suffit d'exécuter l'*Installeur tout-en-un* et le tour est joué !
- Windows Dans une distribution prête à l'emploi *Distribution Python*, basée sur WinPython.
- En tant que *Paquet Wheel* précompilé, qui peut être installé à l'aide de `pip`.
- En tant que *Paquet source*, qui peut être installé à l'aide de `pip` ou manuellement.

### Voir aussi :

Impatient d'essayer la prochaine version de DataLab ? Vous pouvez également installer la dernière version de développement de DataLab à partir de la branche principale du dépôt Git. Voir *Version de développement* pour plus d'informations.

### Paquet Conda

GNU/Linux Windows macOS

Pour installer le paquet `datalab` depuis le canal `conda-forge` (<https://anaconda.org/conda-forge/datalab>), exécutez la commande suivante :

```
$ conda install conda-forge::datalab
```

### Gestionnaire de paquets pip

GNU/Linux Windows macOS

Le paquet `cdl` de DataLab est disponible sur l'index des paquets Python (PyPI) à l'adresse suivante : <https://pypi.python.org/pypi/cdl>.

L'installation de DataLab depuis PyPI avec Qt est aussi simple que d'exécuter cette commande (vous devrez peut-être utiliser `pip3` au lieu de `pip` sur certains systèmes) :

```
$ pip install cdl[qt]
```

Ou, si vous préférez, vous pouvez installer DataLab sans la bibliothèque Qt (non recommandé) :

```
$ pip install cdl
```

---

**Note :** Si vous avez déjà une version antérieure de DataLab installée, vous pouvez la mettre à jour en exécutant la même commande avec l'option `--upgrade` :

```
$ pip install --upgrade cdl[qt]
```

## Installeur tout-en-un

### Windows

DataLab est disponible sous la forme d'une application autonome pour Windows qui ne nécessite pas d'installation de Python. Il suffit d'exécuter l'installateur et vous êtes prêt à partir !

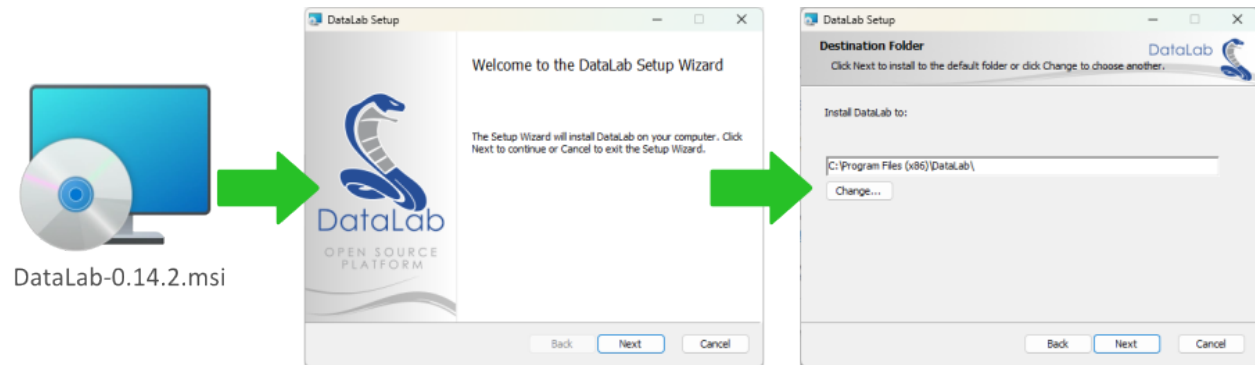


FIG. 1 – Installeur tout-en-un de DataLab pour Windows

Le paquet d'installation est disponible dans la section [Releases](#). Il prend en charge la désinstallation et la mise à jour automatiques (pas besoin de désinstaller DataLab avant d'exécuter l'installateur d'une autre version de l'application).

**Avertissement :** L'installateur Windows de DataLab est disponible pour Windows 7 SP1, 8, 10 et 11.

Sur Windows 7 SP1, avant d'exécuter DataLab (ou toute autre application Python 3), vous devez installer la mise à jour Microsoft *KB2533623 (Windows6.1-KB2533623-x64.msu)* et vous devrez peut-être également installer le [package redistribuable Microsoft Visual C++ 2015-2022](#).

## Distribution Python

### Windows

DataLab est également disponible dans une distribution Python prête à l'emploi, basée sur [WinPython](#). Cette distribution s'appelle [DataLab-WinPython](#) et est disponible dans la section [DataLab-WinPython Releases](#).

La principale différence avec l'installateur tout-en-un est que vous pouvez utiliser la distribution Python à d'autres fins que l'exécution de DataLab, et vous pouvez également l'étendre avec des paquets supplémentaires. En revanche, elle est également *beaucoup plus volumineuse* que l'installateur tout-en-un car elle inclut une distribution Python complète.

**Avertissement :** Alors que l'installateur tout-en-un fournit un paquet monolithique qui garantit la compatibilité de tous ses composants car il ne peut pas être modifié par l'utilisateur, la distribution WinPython est plus flexible et peut donc être endommagée par une mauvaise manipulation de la distribution Python par l'utilisateur. Cela doit être pris en compte lors du choix de la méthode d'installation.



FIG. 2 – DataLab-WinPython est une distribution Python prête à l'emploi incluant la plateforme DataLab.

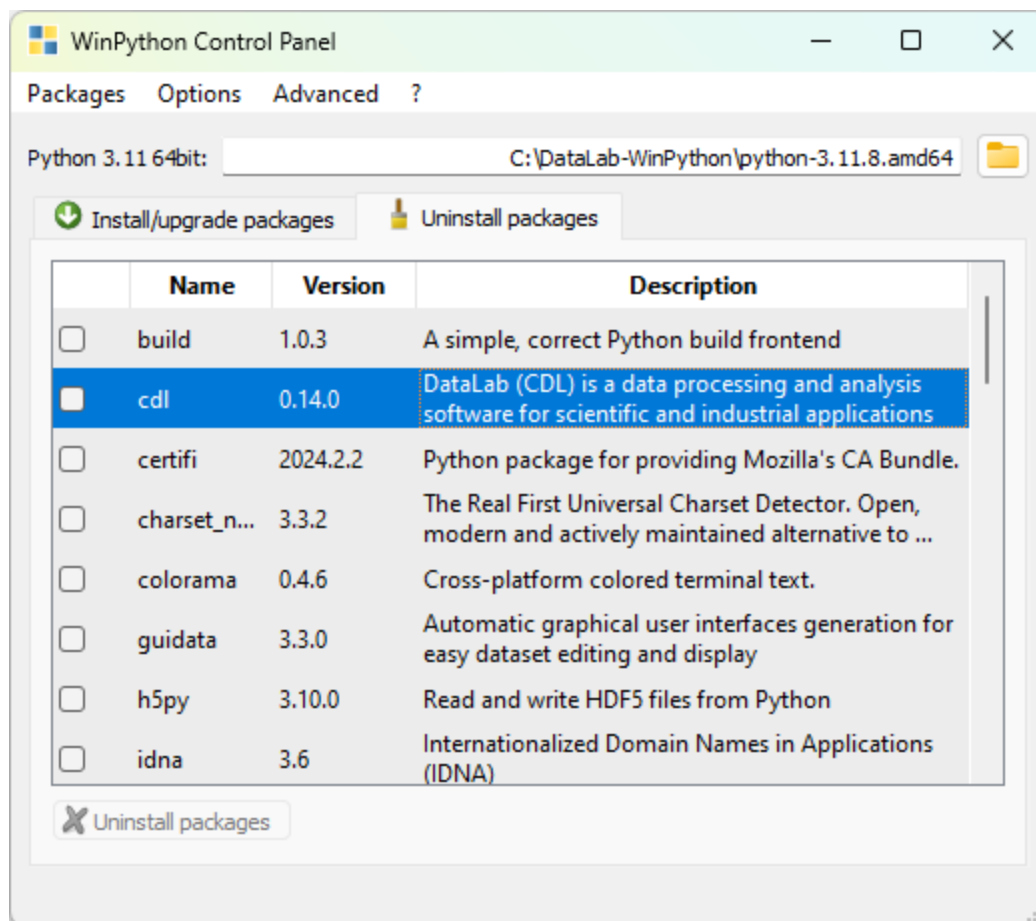


FIG. 3 – Panneau de contrôle DataLab-WinPython

## Paquet Wheel

GNU/Linux Windows macOS

Sur n'importe quel système d'exploitation, l'utilisation de pip et du paquet Wheel est le moyen le plus simple d'installer DataLab sur une distribution Python existante :

```
$ pip install --upgrade DataLab-0.11.1-py2.py3-none-any.whl
```

## Paquet source

GNU/Linux Windows macOS

L'installation de DataLab directement depuis le paquet source peut être effectuée à l'aide de pip :

```
$ pip install --upgrade cdl-0.11.1.tar.gz
```

Ou, si vous préférez, vous pouvez l'installer manuellement en exécutant la commande suivante depuis le répertoire racine du paquet source :

```
$ pip install --upgrade .
```

Enfin, vous pouvez également créer votre propre paquet Wheel et l'installer à l'aide de pip, en exécutant la commande suivante depuis le répertoire racine du paquet source (cela nécessite que les paquets build et wheel soient installés) :

```
$ pip install build wheel # Install build and wheel packages (if needed)
$ python -m build # Build the wheel package
$ pip install --upgrade dist/cdl-0.11.1-py2.py3-none-any.whl # Install the wheel package
```

## Version de développement

GNU/Linux Windows macOS

Si vous souhaitez essayer la dernière version de développement de DataLab, vous pouvez l'installer directement depuis la branche principale du dépôt Git.

La première fois que vous installez DataLab depuis le dépôt Git, entrez la commande suivante :

```
$ pip install git+https://github.com/DataLab-Platform/DataLab.git
```

Ensuite, si vous souhaitez à un moment donné passer à la dernière version de DataLab, exécutez simplement la même commande avec les options pour forcer la réinstallation du paquet sans gérer les dépendances (car cela réinstallerait toutes les dépendances) :

```
$ pip install --force-reinstall --no-deps git+https://github.com/DataLab-Platform/
↪DataLab.git
```

**Note :** Si les dépendances ont changé, vous devrez peut-être exécuter la même commande que ci-dessus, mais sans l'option `--no-deps`.

## 1.1.2 Dépendances

**Note :** L'installateur tout-en-un de DataLab inclut déjà toutes ces dépendances ainsi que Python lui-même.

Le paquet `cdl` requiert les modules Python suivants :

Nom	Version	Description
Python	>=3.9, <4	Langage de programmation Python
guidata	>= 3.7	Génération automatique d'interfaces graphiques pour l'édition et l'affichage de jeux de données
PlotPy	>= 2.7	Outils de tracé de courbes et d'images pour les applications Python/Qt
SciPy	>= 1.5, < 1.15.0	Algorithmes fondamentaux pour le calcul scientifique en Python
scikit-image	>= 0.18	Traitement d'images en Python
pandas	>= 1.2	Analyse de données, séries temporelles et statistiques
PyWavelets	>= 1.1	PyWavelets, module de transformation en ondelettes
psutil	>= 5.7	Cross-platform lib for process and system monitoring in Python. NOTE : the syntax of this script MUST be kept compatible with Python 2.7.
PyQt5	>=5.11	Bibliothèque d'interfaces graphiques Qt pour Python

Modules optionnels pour le développement :

Nom	Version	Description
ruff		Analyseur de code et formateur Python extrêmement rapide, écrit en Rust.
pylint		Analyseur statique de code Python
Coverage		Mesure de la couverture de code pour Python
pyinstaller	>=6.0	PyInstaller permet de créer un exécutable unique à partir d'une application Python et de ses dépendances.

Modules optionnels pour la génération de la documentation :

Nom	Version	Description
PyQt5		Bibliothèque d'interfaces graphiques Qt pour Python
sphinx		Générateur de documentation Python
sphinx_intl		Utilitaire pour la traduction de la documentation générée par Sphinx.
sphinx-sitemap		Sitemap generator for Sphinx
myst_parser		Un parseur étendu compatible avec [CommonMark]( <a href="https://spec.commonmark.org/">https://spec.commonmark.org/</a> )
sphinx_design		Extension sphinx pour la conception de composants web réactifs.
sphinx-copybutton		Extension sphinx ajoutant un bouton de copie à chaque cellule de code.
pydata-sphinx-theme		Thème Bootstrap pour Sphinx de la communauté PyData

Modules optionnels pour l'exécution de la suite de tests :

Nom	Version	Description
pytest		pytest : tests simples et puissants avec Python
pytest-xvfb		Plugin pytest pour exécuter Xvfb (ou Xephyr/Xvnc) pour les tests.

---

**Note :** Python 3.11 et PyQt5 sont les références pour la version de production

---

## 1.2 Cas d'usage, principales fonctionnalités et points forts

DataLab est une plateforme de traitement et de visualisation de données (signaux ou images) qui intègre de nombreuses fonctions. Développé en Python, il bénéficie de la richesse de l'écosystème associé en termes de bibliothèques scientifiques et techniques.

### 1.2.1 Quelles sont les applications de DataLab ?

#### Exemples concrets

Quelques exemples concrets et spécifiques illustrent la nature des travaux pouvant être réalisés avec DataLab :

- Traitement de données expérimentales (signaux et images) acquises sur une installation scientifique dans le domaine nucléaire
- Traitement de données acquises par un capteur dans un contexte industriel
- Traitement d'images acquises par une caméra dans un contexte médical
- Détection automatique de défauts sur une surface, dans le contexte du contrôle qualité
- Détection automatique des positions des tâches laser dans une scène, dans le contexte de l'alignement laser
- Alignement d'instrument par traitement d'image
- Détection automatique de motifs et corrections géométriques d'images, dans le contexte du contrôle non destructif

#### Modes d'utilisation

Selon l'application, DataLab peut être utilisé selon trois modes différents :

- **Mode autonome** : DataLab est une application de traitement à part entière qui peut être adaptée aux besoins du client par l'ajout de plugins spécifiques à son métier.
- **Mode embarqué** : DataLab est intégré dans votre application pour apporter les fonctionnalités de traitement et de visualisation nécessaires.
- **Mode commandé à distance** : DataLab communique avec votre application, lui permettant de bénéficier de ses fonctionnalités sans perturber l'expérience utilisateur.

## Cas d'usage

### Voir aussi :

Pour des exemples pratiques de cas d'usage, voir la section *Tutoriels* :

- La plupart des tutoriels décrivent des exemples concrets d'utilisation de DataLab dans un contexte scientifique ou technique.
- Concernant l'utilisation de DataLab avec un IDE (Integrated Development Environment) tel que Visual Studio Code ou Spyder, voir le tutoriel *DataLab et Spyder : un mariage parfait*.
- Quant à l'utilisation de DataLab avec des notebooks Jupyter, c'est l'un des sujets abordés dans le tutoriel *Ajouter vos propres fonctionnalités*.

DataLab est un outil polyvalent qui peut être utilisé dans différents contextes :

### Traitement de données

DataLab est un outil puissant pour le traitement de signaux et d'images. Il peut être utilisé pour développer des algorithmes complexes, ou pour prototyper rapidement une chaîne de traitement.

Voir nos *Tutoriels* pour des exemples pratiques d'utilisation dans le traitement de données.

### Outil de soutien aux travaux scientifiques/techniques

DataLab peut être utilisé comme un outil de soutien aux travaux scientifiques/techniques. Il vous permet de visualiser et de traiter des données, et de partager vos résultats avec vos collègues. Il peut facilement être adapté à vos besoins par l'ajout de plugins, et peut même être utilisé avec vos outils quotidiens (par exemple Visual Studio Code, Spyder, ... ou des notebooks Jupyter).

Voir nos *Tutoriels* pour des exemples pratiques d'utilisation dans un contexte scientifique/technique.

### Prototypage d'une application de traitement de données

DataLab peut être utilisé pour prototyper rapidement une application de traitement de données. Il peut ensuite être utilisé comme base pour le développement d'une application à part entière.

Voir le tutoriel *Prototypage d'une chaîne de traitement personnalisée* pour un exemple concret.

### Débogage d'une application de traitement de données

DataLab peut être utilisé comme un outil de débogage avancé pour vos applications de traitement de données, indépendamment de l'environnement de développement ou du langage utilisé (Python, C#, C++, ...). Tout ce dont vous avez besoin est de pouvoir communiquer avec DataLab via son interface de pilotage à distance (protocole XML-RPC standard). Cela vous permet d'envoyer des données à DataLab (signaux, images ou même des formes géométriques), de visualiser les données à chaque étape de la chaîne de traitement, de les manipuler pour mieux comprendre le comportement de vos algorithmes, et même de les modifier pour tester la robustesse de votre code.

Voir le tutoriel *Déboguer votre algorithme avec DataLab* pour un aperçu rapide de cette fonctionnalité.

---

**Note :** DataLab peut également être piloté depuis votre environnement de développement familier (par exemple Visual Studio Code, Spyder, ...) ou depuis un notebook Jupyter, afin d'effectuer des calculs en utilisant vos fonctions de traitement tout en bénéficiant des fonctionnalités avancées de DataLab. Voir les tutoriels *Prototypage d'une chaîne de traitement personnalisée* ou *DataLab et Spyder : un mariage parfait* pour des exemples d'utilisation.

---

Avec son expérience utilisateur conviviale et ses modes d'utilisation polyvalents, DataLab permet un développement efficace de vos applications de traitement et de visualisation de données tout en bénéficiant d'une plateforme technologique industrielle.

## 1.2.2 Principales fonctionnalités

Les principales fonctionnalités techniques de DataLab sont :

- Prise en charge de nombreux formats de données standards et propriétaires
- Ouverture d'un nombre arbitraire d'objets (signaux ou images) pour un traitement par lot, avec possibilité de définir des groupes d'objets
- Visualisation simultanée de plusieurs objets avec prise en charge des annotations
- Opérations et traitements standards sur les signaux et les images
- Traitement d'image avancé (restauration, morphologie, détection de contours, etc.)
- Gestion de plusieurs régions d'intérêt (calculs, extractions)
- Éditeur de macro-commandes
- API pilotable à distance
- Console Python interactive embarquée

## 1.2.3 Points forts

Pour résumer, les quatre points forts de DataLab sont les suivants :

### Extensibilité

Le système de plugins de DataLab permet de coder facilement de nouvelles fonctionnalités (traitement spécifique, formats de fichiers spécifiques, interfaces graphiques personnalisées). Il peut également être utilisé comme une plateforme personnalisable.

### Interopérabilité

DataLab peut également être intégré dans votre propre application. Par exemple, au sein de logiciels de traitement de données, de systèmes de contrôle de niveau machine ou d'applications de banc d'essai.

### Automatisation

Une API publique de haut niveau permet de piloter à distance DataLab pour ouvrir et traiter des données.

### Maintenabilité et testabilité

DataLab est un logiciel de traitement scientifique et technique industriel. Les tests automatisés intégrés à DataLab couvrent 90 % de ses fonctionnalités, ce qui est significatif pour un logiciel avec des interfaces graphiques et permet de réduire les risques de régression. De plus, la suite de tests comprend des tests de validation basés soit sur des données de référence, soit sur des solutions analytiques

### Voir aussi :

Voir la section [Validation](#) pour plus d'informations sur la stratégie de validation de DataLab.

## 1.3 Fonctionnalités principales

Cette page présente brièvement les principales fonctionnalités de DataLab.

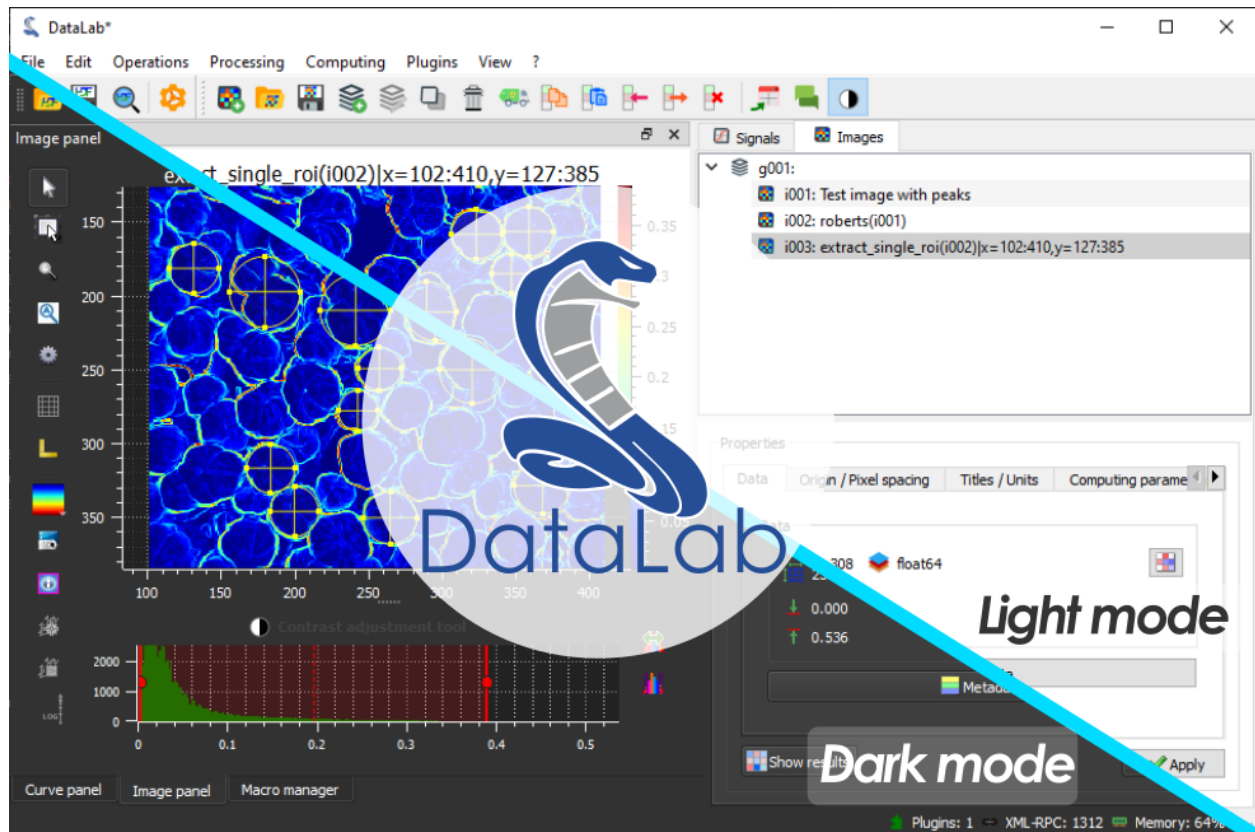


FIG. 4 – DataLab prend en charge le mode sombre et le mode clair en fonction des paramètres de votre plateforme (cela est géré par le paquet `guidata` et peut être remplacé en définissant la variable d'environnement `QT_COLOR_MODE` sur `dark` ou `light`).

### 1.3.1 Visualisation de données

Signal	Image	Fonctionnalité
✓	✓	Captures d'écran (enregistrer, copier)
✓	Axe Z	Échelles linéaire/logarithmique
✓	✓	Édition de table de données
✓	✓	Statistiques sur ROI défini par l'utilisateur
✓	✓	Marqueurs
	✓	Ratio d'aspect (1 :1, personnalisé)
	✓	Plus de 50 échelles de couleurs disponibles (personnalisables)
	✓	Profils d'intensité (ligne, moyen, radial)
✓	✓	Annotations
✓	✓	Persistance des paramètres dans l'espace de travail
	✓	Distribuer les images sur une grille
✓	✓	Vues simples ou superposées

### 1.3.2 Traitement de données

Signal	Image	Fonctionnalité
✓	✓	Processus isolé pour l'exécution des calculs
✓	✓	Contrôle à distance depuis Jupyter, Spyder ou tout autre IDE
✓	✓	Contrôle à distance depuis une application tierce
✓	✓	Somme, moyenne, différence, produit, ...
✓	✓	Opérations avec une constante
✓	✓	Extraction de ROI, permutation des axes X/Y
✓		Détection de pics semi-automatique
✓		Convolution
	✓	Correction de champ plat
	✓	Rotation (retournement, rotation), redimensionnement, ...
	✓	Profils d'intensité (ligne, moyen, radial)
	✓	Binning de pixels
✓	✓	Racine carrée, puissance, logarithme, exponentielle, ...
✓		Dérivée, intégrale
✓	✓	Calibration linéaire
✓	✓	Normalisation, rognage, correction de décalage
✓		Inversion de l'axe X
	✓	Seuillage (manuel, Otsu, ...)
✓	✓	Filtre gaussien, filtre de Wiener
✓	✓	Moyenne mobile, médiane mobile
✓	✓	FFT, FFT inverse, spectre de puissance/phase/magnitude, densité spectrale de puissance
✓		Interpolation, rééchantillonnage
✓		Élimination de tendance
✓		Ajustement interactif : Gauss, Lorentz, Voigt, polynôme, CDF, ...
✓		Ajustement interactif multi-gaussien
✓		Filtres fréquentiels (passe-bas, passe-haut, passe-bande, coupe-bande)
✓		Fenêtrage (Hamming, Hanning, ...)
	✓	Filtre de Butterworth
	✓	Correction d'exposition (gamma, log, ...)
	✓	Restauration (variation totale, bilatérale, ...)

suite sur la page suivante

Tableau 1 – suite de la page précédente

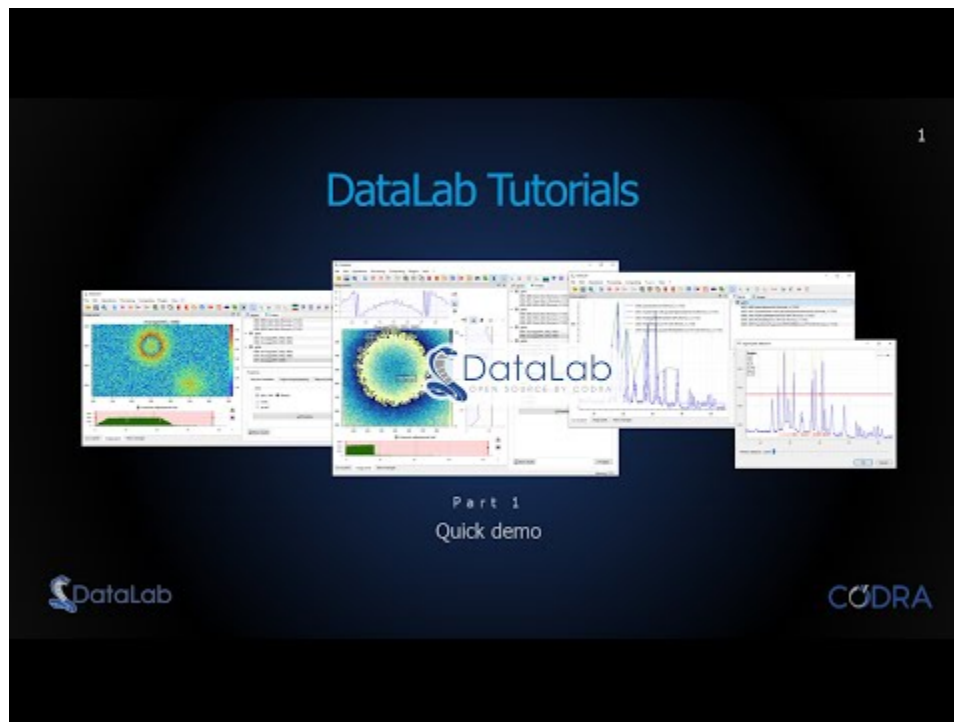
Signal	Image	Fonctionnalité
	✓	Morphologie (érosion, dilatation, ...)
	✓	Détection de contours (Roberts, Sobel, ...)
✓	✓	Analyse sur ROI personnalisée
✓		FWHM, FW @ $1/e^2$
✓		Paramètres dynamiques (ENOB, SNR, ...), Période/Taux d'échantillonnage
	✓	Barycentre (méthode robuste par rapport au bruit)
	✓	Centre du cercle d'encadrement minimal
	✓	Détection de pics 2D
	✓	Détection de contours
	✓	Transformée de Hough circulaire
	✓	Détection de taches (OpenCV, Laplacien de Gauss, ...)

## 1.4 Tutoriels

### 1.4.1 Tutoriels vidéo

Les tutoriels vidéo de DataLab ont pour but de fournir un matériel complémentaire à la documentation et aux autres tutoriels disponibles ici.

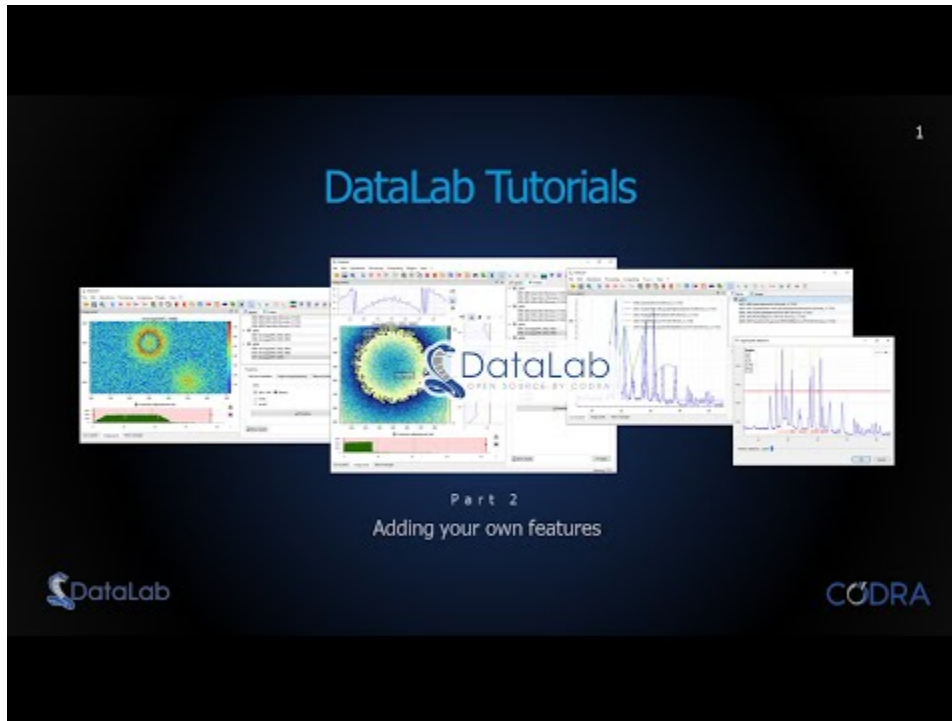
#### Démo rapide



Dans cette première vidéo, nous allons rapidement passer en revue les principales fonctionnalités de DataLab, et montrer comment effectuer un traitement de signal et d'image de base.

**Avertissement :** Cette vidéo est volontairement courte et ne couvre pas toutes les fonctionnalités de DataLab. Elle a pour but de vous donner un aperçu rapide du logiciel. Pour une description plus détaillée des fonctionnalités, veuillez regarder les autres vidéos ou lire la documentation.

## Ajouter vos propres fonctionnalités



Dans ce tutoriel, nous allons montrer comment ajouter vos propres fonctionnalités à DataLab en utilisant trois approches différentes :

1. Les macro-commandes, en utilisant le gestionnaire de macro intégré
2. Le contrôle à distance de DataLab depuis un IDE externe (par exemple Spyder) ou un notebook Jupyter
3. Les plugins

Le premier point commun entre ces trois approches est qu'elles reposent toutes sur l'API de haut niveau de DataLab, qui permet d'interagir avec presque tous les aspects du logiciel. Cette API est ici accédée à l'aide de scripts Python, mais elle peut également être accédée à l'aide de n'importe quel autre langage lors de l'utilisation de l'approche de contrôle à distance (car elle repose sur un protocole de communication standard, XML-RPC).

Le second point commun est qu'ils utilisent tous du code Python et des objets proxy compatibles, de sorte que le même code peut être au moins partiellement réutilisé dans les trois approches.

## 1.4.2 Autres tutoriels

Les tutoriels suivants sont des guides détaillés étape par étape pour effectuer des tâches spécifiques avec DataLab, ou pour illustrer les fonctionnalités du logiciel dans le contexte d'un problème scientifique ou technique. Chaque tutorial se concentre sur un aspect spécifique du logiciel et est destiné à être autonome.

### Traitement d'un spectre

Cet exemple montre comment traiter un spectre avec DataLab :

- Lire le spectre à partir d'un fichier
- Appliquer un filtre au spectre
- Extraire une région d'intérêt
- Ajuster un modèle au spectre
- Sauvegarder l'espace de travail dans un fichier

Tout d'abord, nous ouvrons DataLab et lisons le spectre à partir d'un fichier.

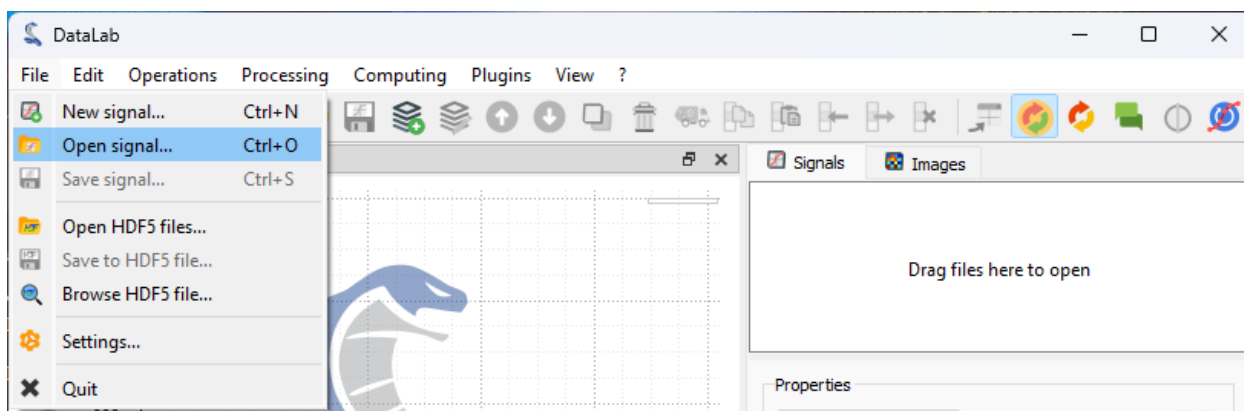


FIG. 5 – Ouvrir le fichier de spectre avec « Fichier > Ouvrir... », ou avec le bouton dans la barre d'outils, ou en faisant glisser et déposer le fichier dans DataLab (sur le panneau de droite).

Ici, nous générons en fait le signal à partir d'un fichier de données de test (en utilisant « Plugins > Test data > Load spectrum of paracetamol »), mais le principe est le même.

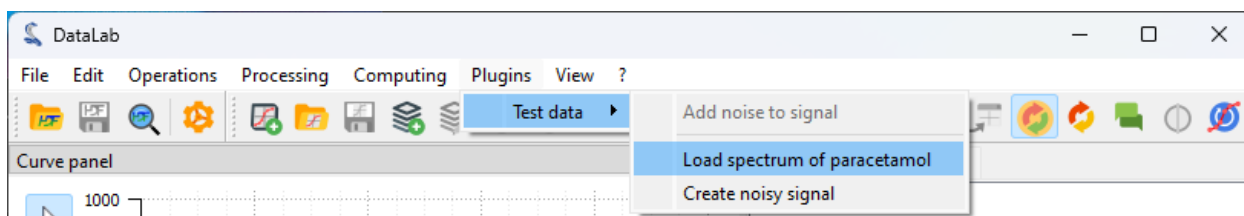


FIG. 6 – Utiliser le plugin « Données de test » est un moyen pratique de générer des données de test pour les tutoriels, mais vous pouvez utiliser n'importe quel fichier contenant un spectre, comme un spectre provenant d'une véritable expérience.

Le spectre est affiché dans la fenêtre principale.

A présent, traitons ce spectre en lui appliquant un filtre. Nous utiliserons un filtre de Wiener, qui est un filtre qui peut être utilisé pour supprimer le bruit d'un signal, même si cela paraît superflu dans le cas présent.

Si nous voulons analyser une région spécifique du spectre, nous pouvons l'extraire du spectre en utilisant la fonction « Extraction de ROI » du menu « Opérations ».

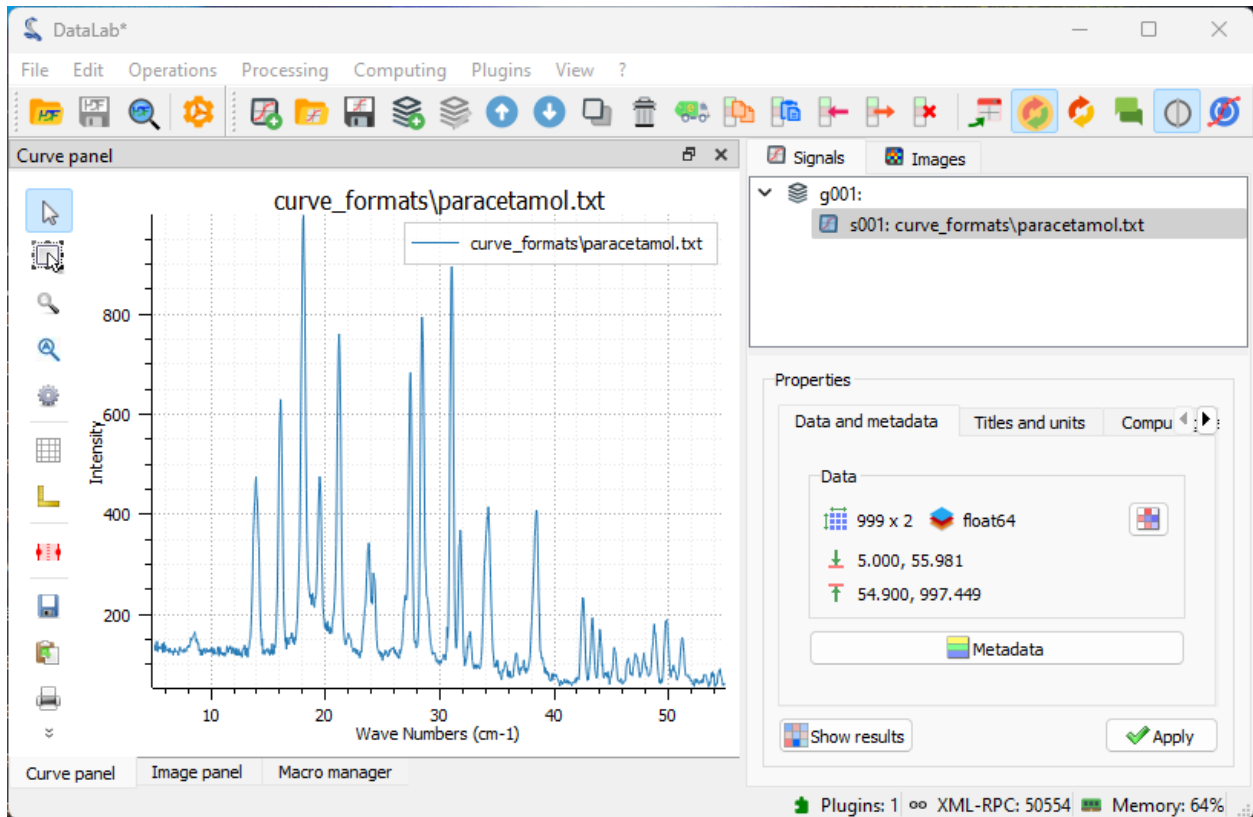


FIG. 7 – Le spectre est un signal 1D, il est donc affiché sous forme de courbe. L'axe horizontal est l'axe de l'énergie et l'axe vertical est l'axe de l'intensité.

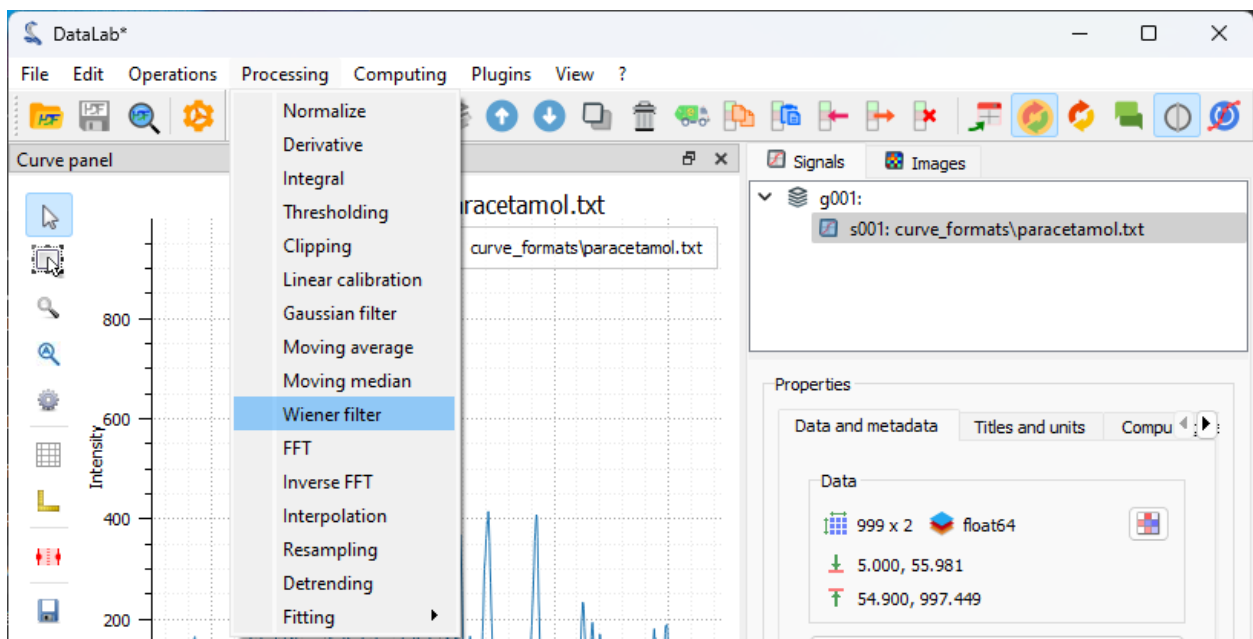


FIG. 8 – Ouvrir la fenêtre de filtre avec « Traitement > Filtre de Wiener ».

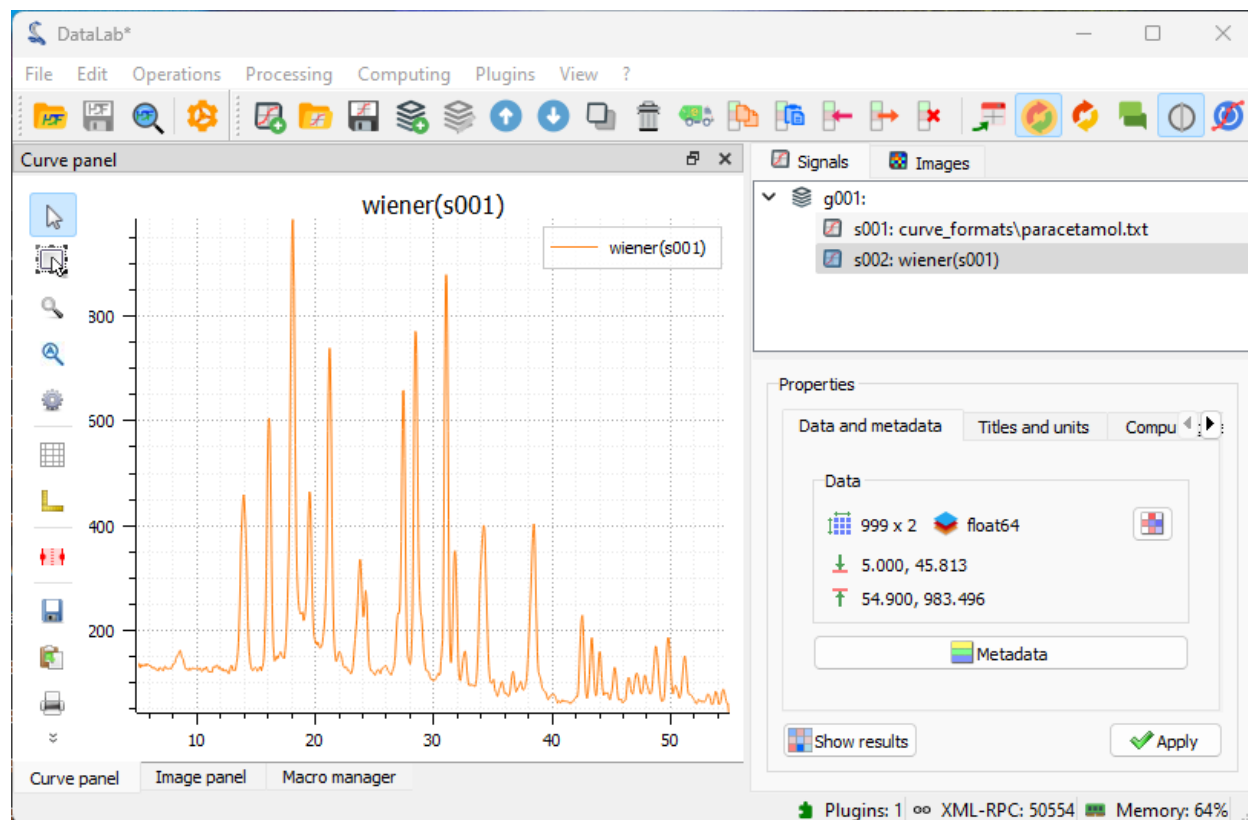


FIG. 9 – Le résultat du filtre est affiché dans la fenêtre principale.

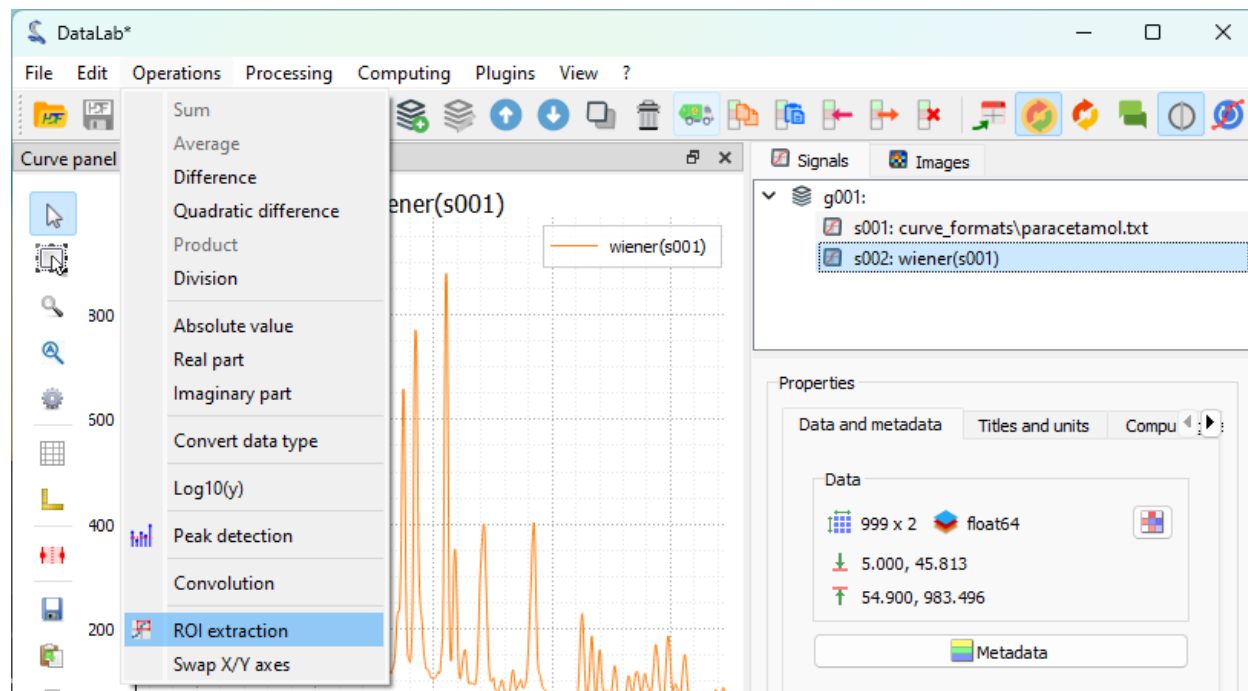


FIG. 10 – Ouvrir la fenêtre d'extraction de ROI avec « Opérations &gt; Extraction de ROI ».

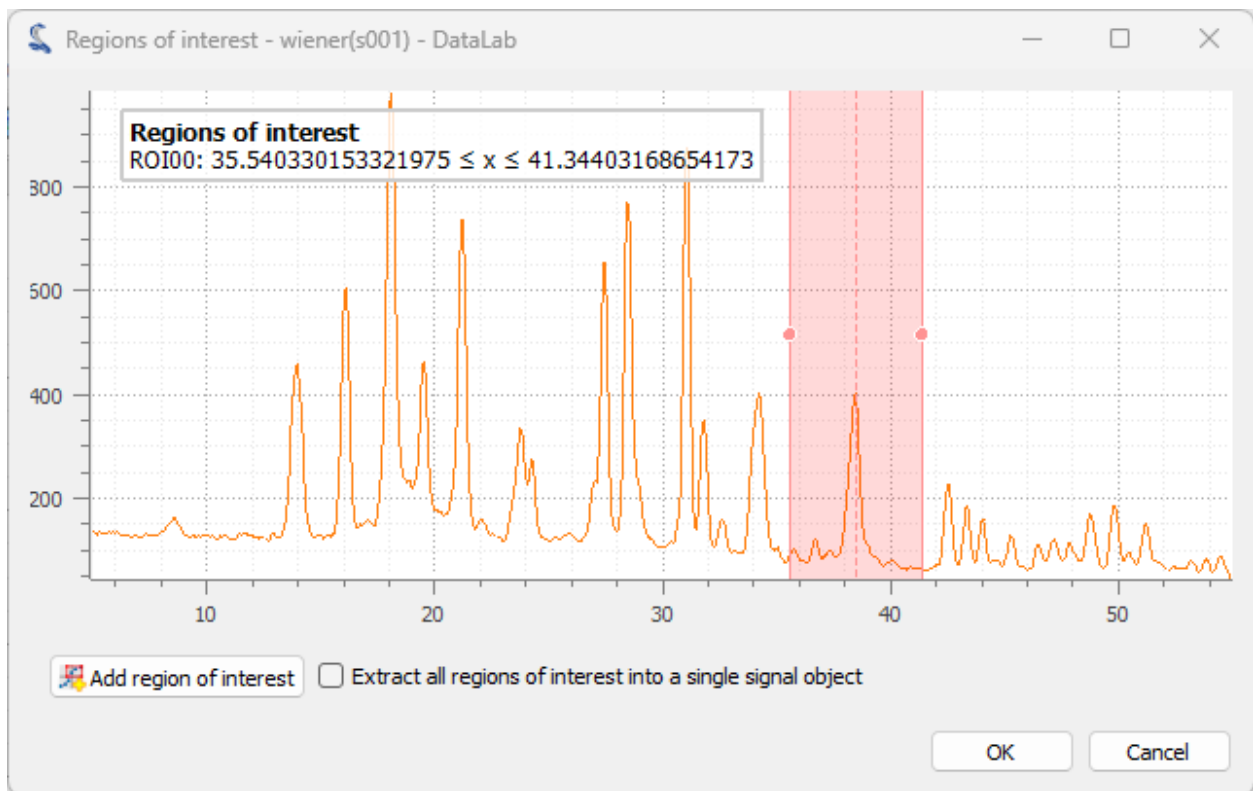


FIG. 11 – La boîte de dialogue « Régions d'intérêt » est affichée. Cliquez sur « Ajouter une ROI » et redimensionnez la fenêtre horizontale pour sélectionner la zone. Ensuite, cliquez sur « OK ».

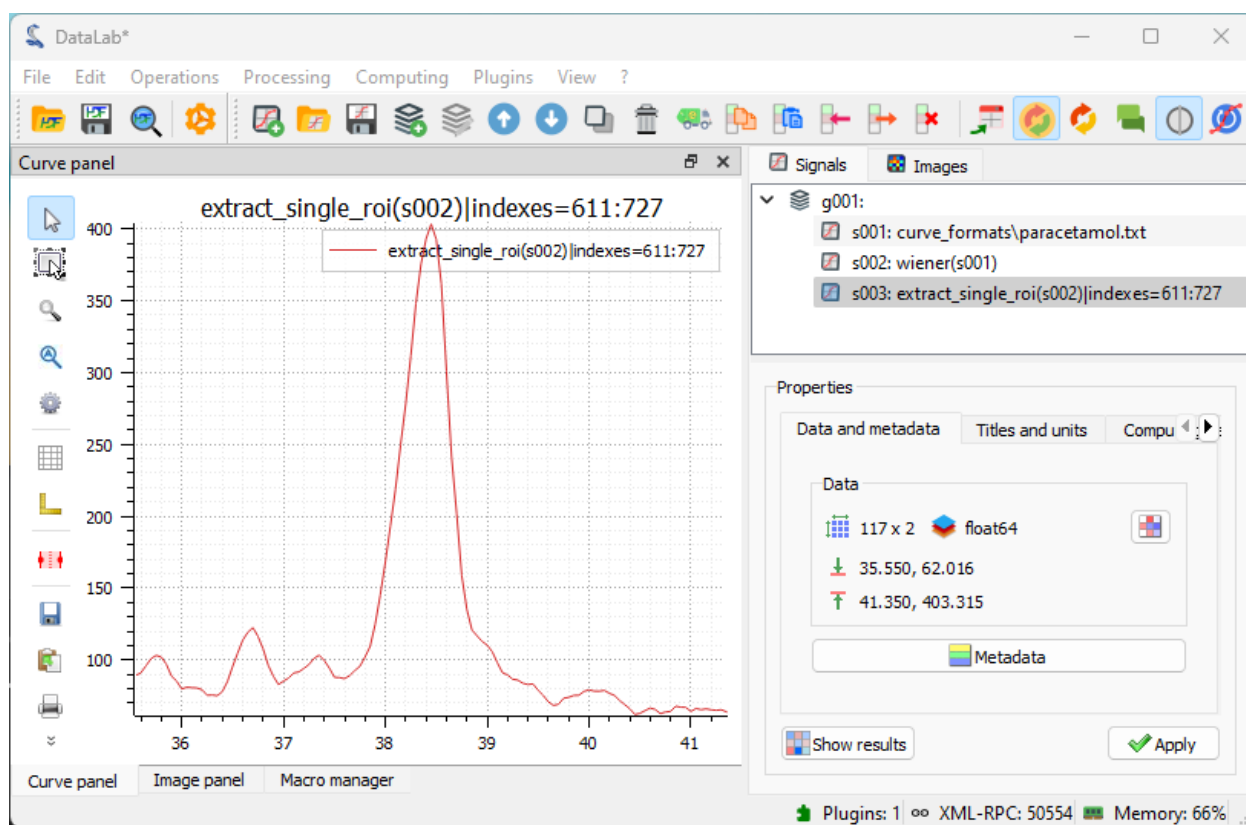


FIG. 12 – La région d'intérêt est affichée dans la fenêtre principale.

Essayons d'ajuster un modèle au spectre. Nous utiliserons un modèle gaussien, qui est un modèle typiquement utilisé pour ajuster un pic dans un spectre.

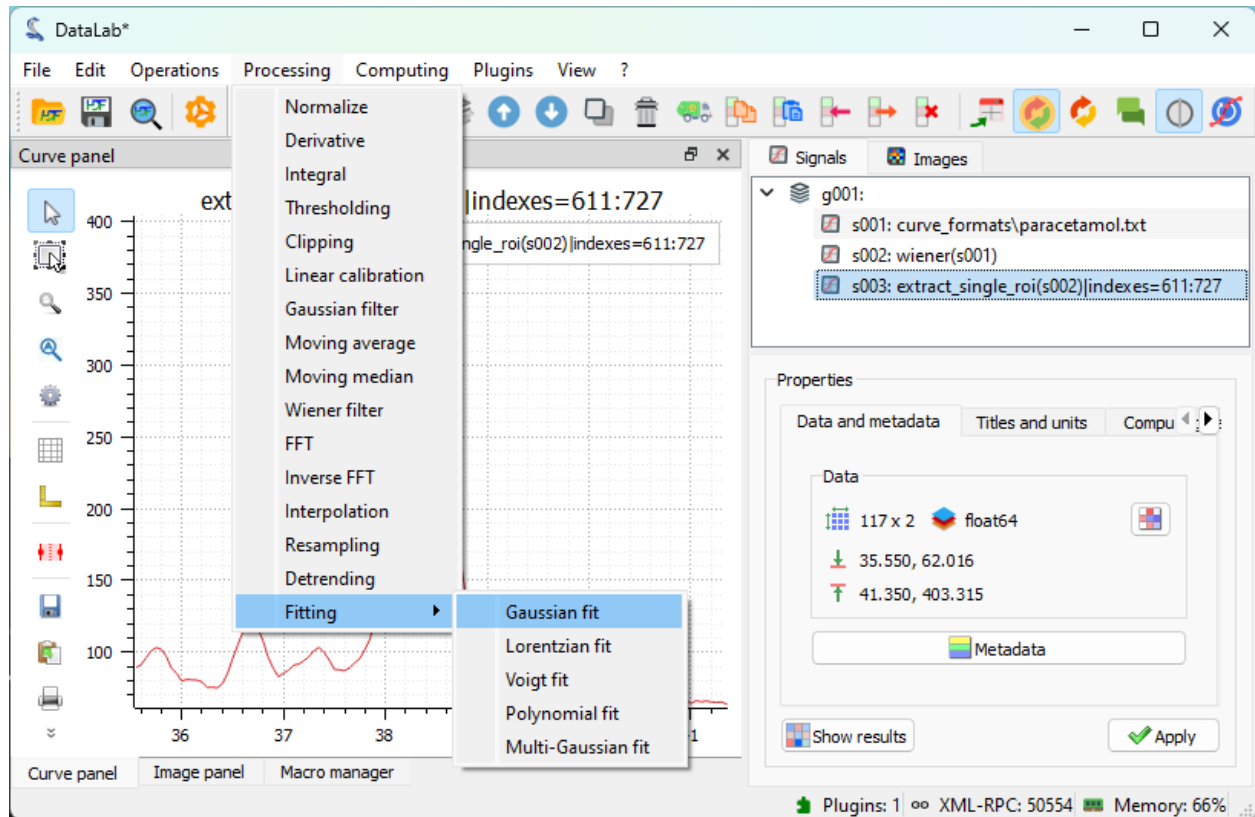


FIG. 13 – Ouvrir la fenêtre d'ajustement du modèle avec « Traitement > Ajustement > Ajustement gaussien ».

Pour faire la démonstration d'une autre fonction de traitement, nous pouvons également essayer la fonctionnalité d'élimination de tendance.

Lors de l'analyse d'un spectre, il peut être utile d'essayer d'identifier les pics dans le spectre. Nous pouvons le faire en ajustant un modèle multi-gaussien au spectre, en utilisant la fonction « Traitement > Ajustement > Ajustement multi-gaussien ».

Nous aurions également pu utiliser la fonction « Détection de pics » du menu « Opérations » pour détecter les pics dans le spectre.

Enfin, nous pouvons sauvegarder l'espace de travail dans un fichier. L'espace de travail contient tous les signaux qui ont été chargés dans DataLab, ainsi que les résultats du traitement. Il contient également les paramètres de visualisation (couleurs de courbe, etc.).

Si vous voulez charger à nouveau l'espace de travail, vous pouvez utiliser le « Fichier > Ouvrir un fichier HDF5... » (ou le bouton dans la barre d'outils) pour charger l'ensemble de l'espace de travail, ou le « Fichier > Parcourir un fichier HDF5... » (ou le bouton dans la barre d'outils) pour charger uniquement une sélection d'ensembles de données de l'espace de travail.

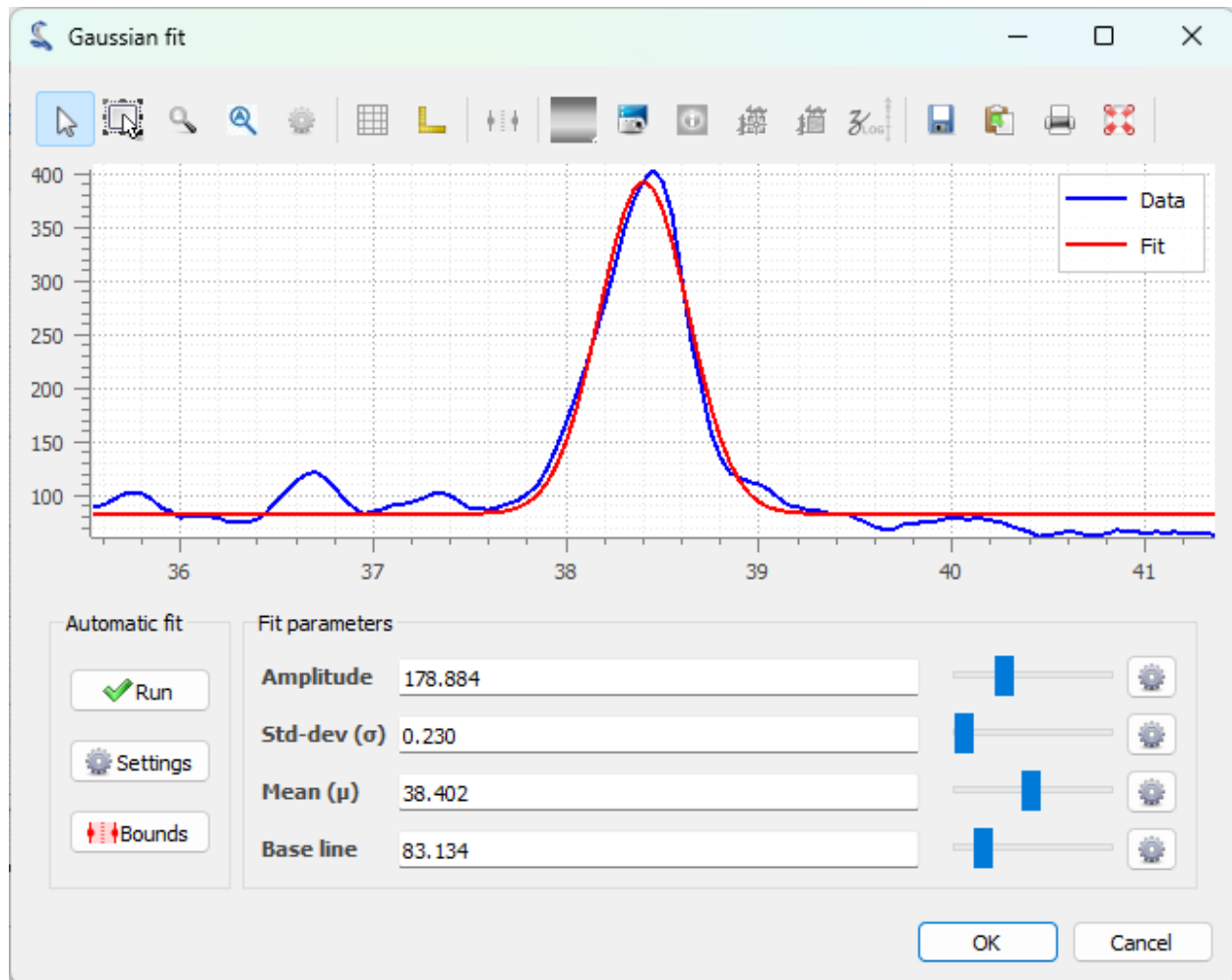


FIG. 14 – La boîte de dialogue « Ajustement gaussien » est affichée. Un ajustement automatique est effectué par défaut. Cliquez sur « OK » (ou essayez éventuellement d'ajuster le modèle manuellement en ajustant les paramètres ou les curseurs, ou essayez de modifier les paramètres d'ajustement automatique).

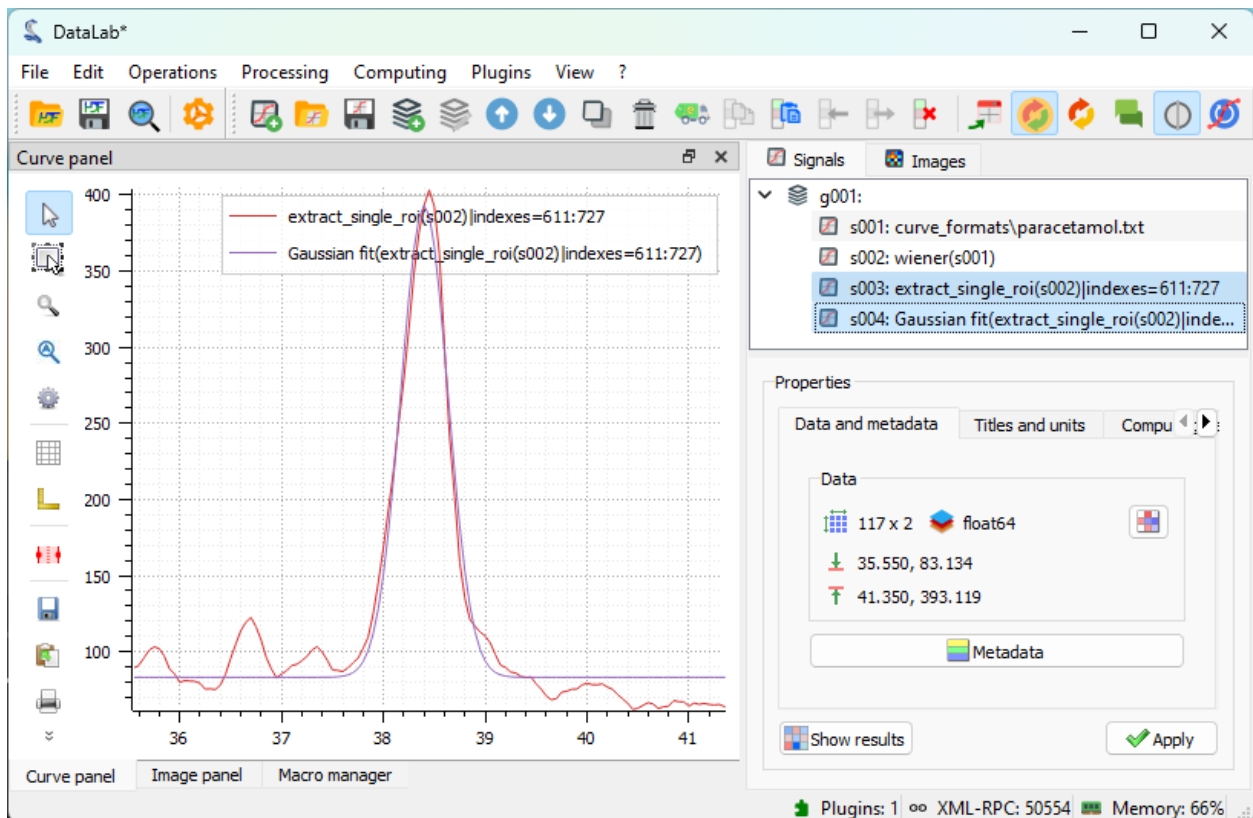


FIG. 15 – Le résultat de l’ajustement est affiché dans la fenêtre principale. Ici, nous avons sélectionné à la fois le spectre et l’ajustement dans le panneau « Signaux » à droite, donc les deux sont affichés dans le panneau de visualisation à gauche.

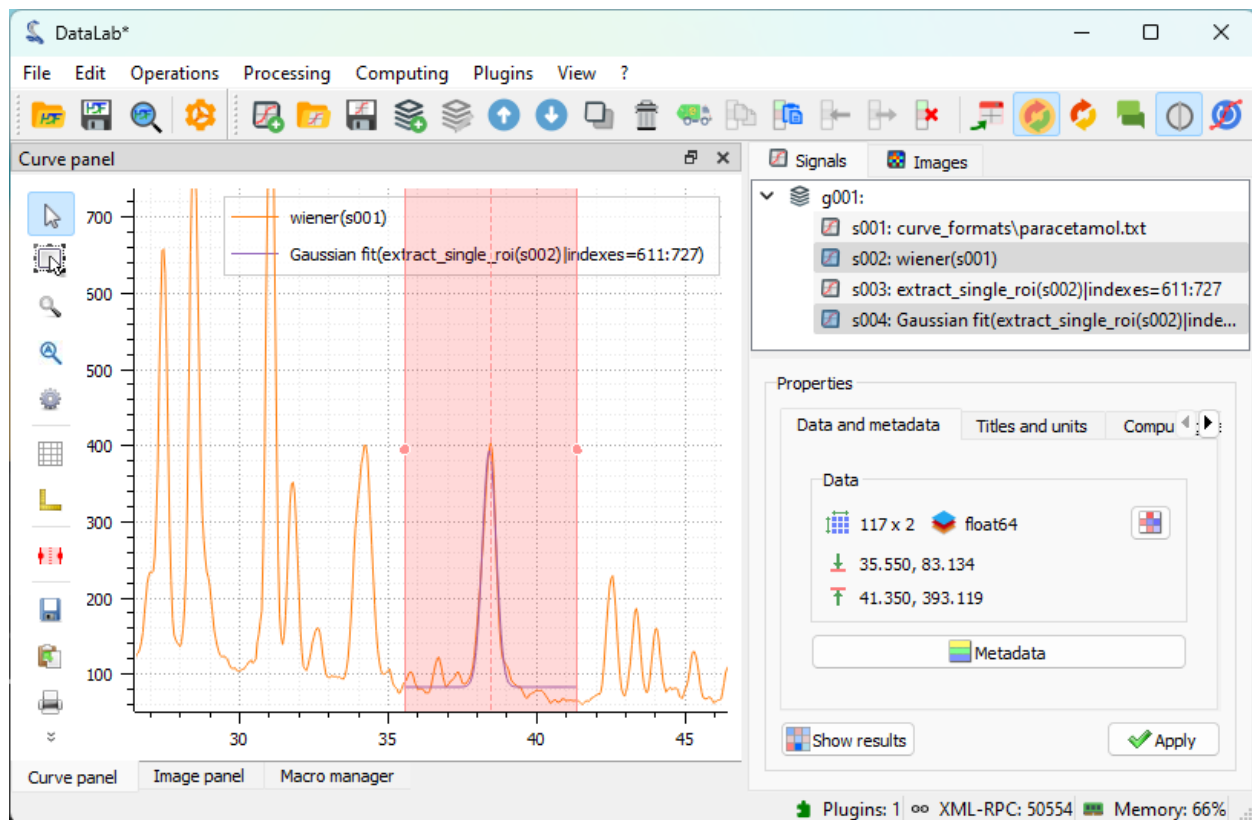


FIG. 16 – Nous pouvons également sélectionner le spectre complet et l’ajustement dans le panneau « Signaux » à droite, de sorte que les deux soient affichés dans le panneau de visualisation à gauche, si cela a un sens pour l’analyse que nous voulons effectuer. Notez que la visualisation du spectre complet contient également la région d’intérêt que nous avons extraite précédemment.

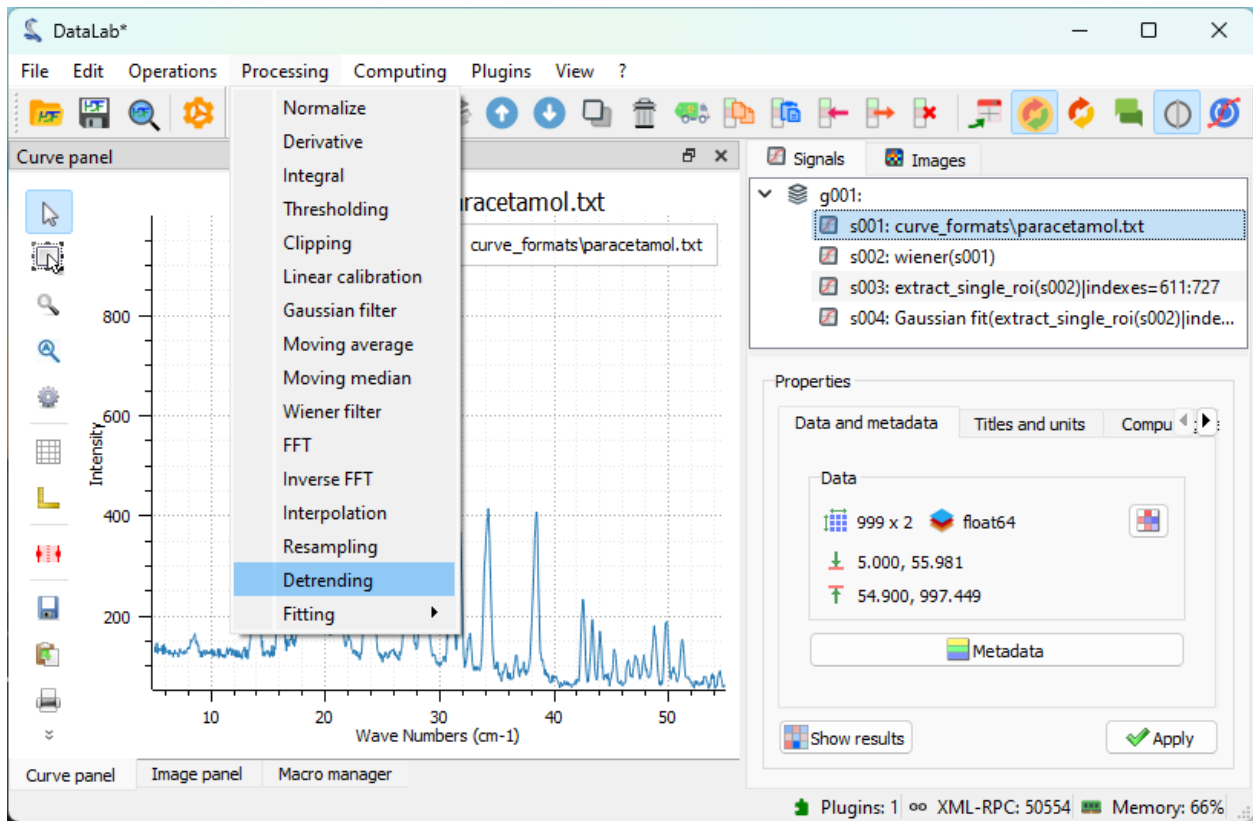


FIG. 17 – Exécutez la fonction « Traitement > Elimination de tendance ».

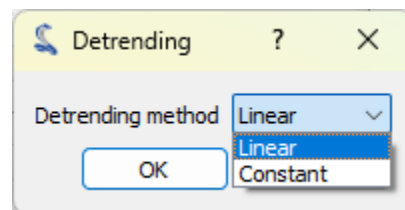


FIG. 18 – Nous choisissons une méthode d'élimination de tendance linéaire, et nous cliquons sur « OK ».

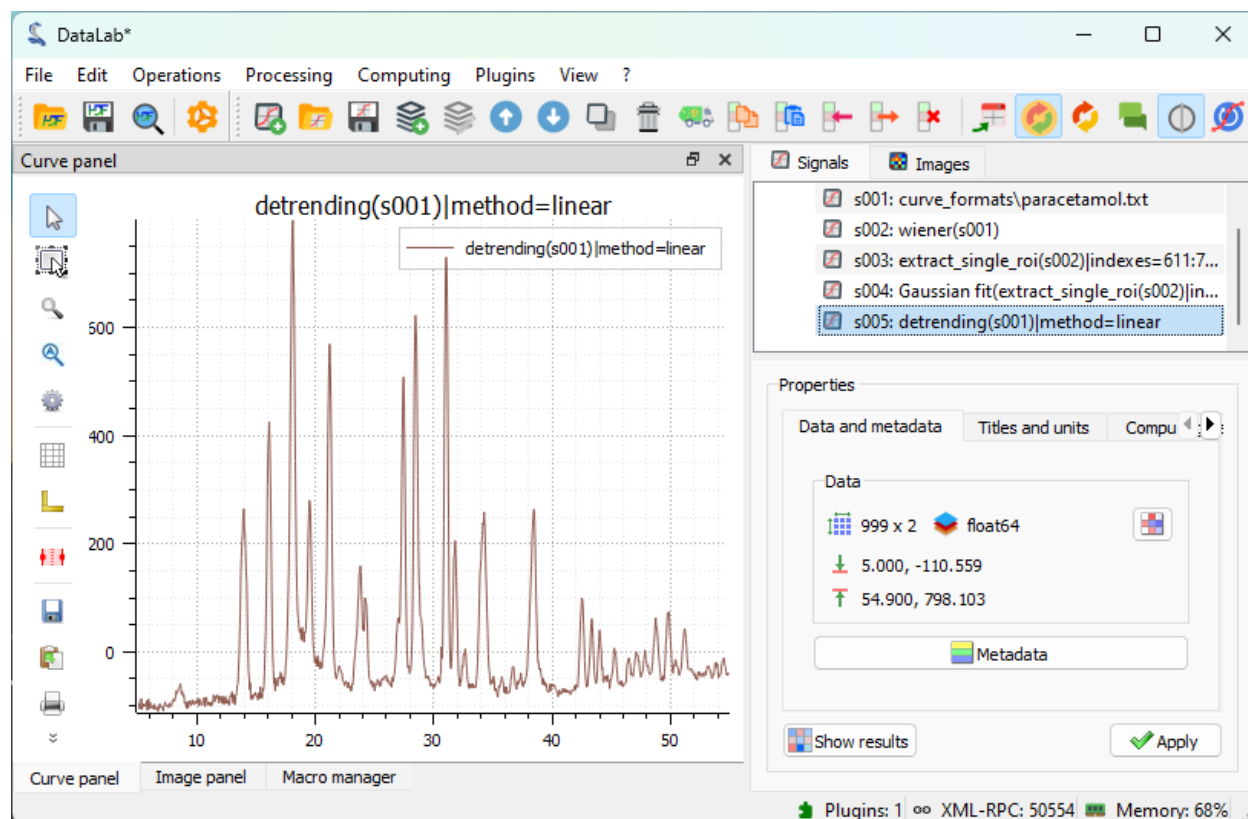


FIG. 19 – Le résultat du traitement est affiché dans la fenêtre principale (dans ce cas précis, l'élimination de tendance n'est pas nécessairement appropriée, mais c'est juste pour démontrer la fonctionnalité).

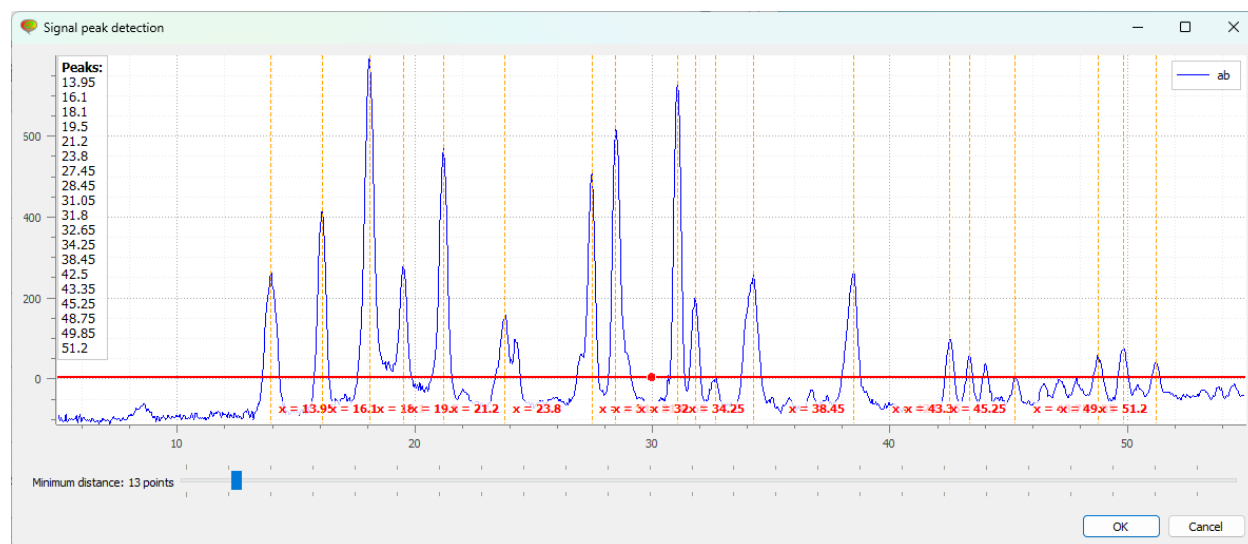


FIG. 20 – Tout d'abord, une boîte de dialogue « Détection de pics de signal » est affichée. Nous pouvons ajuster la position du curseur vertical pour sélectionner le seuil de détection des pics, ainsi que la distance minimale entre deux pics. Ensuite, nous cliquons sur « OK ».

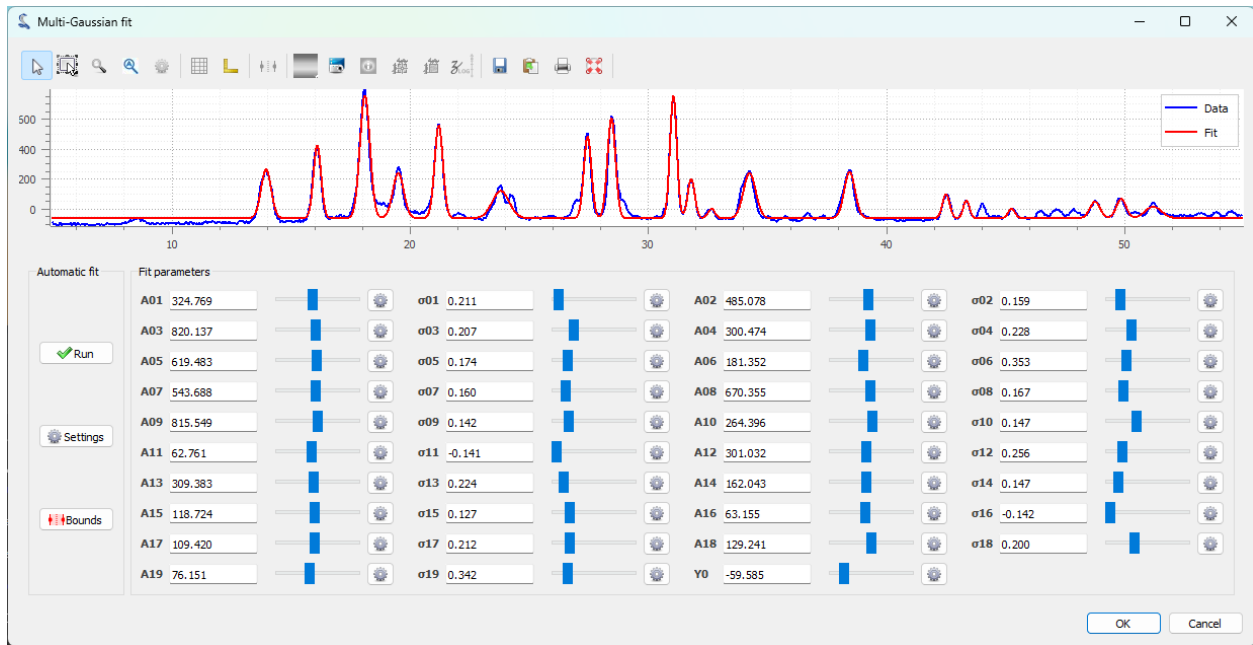


FIG. 21 – La boîte de dialogue « Ajustement multi-gaussien » est affichée. Un ajustement automatique est effectué par défaut. Cliquez sur « OK » (ou essayez éventuellement d’ajuster le modèle manuellement en ajustant les paramètres ou les curseurs, ou essayez de modifier les paramètres d’ajustement automatique).

## Détection de taches sur une image

Cet exemple montre comment détecter des taches sur une image avec DataLab, et couvre également d’autres fonctionnalités telles que le système de plugins :

- Ajouter un nouveau plugin à DataLab
- Débruitage d’une image
- Détecter des taches sur une image
- Enregistrer l’espace de travail dans un fichier

Tout d’abord, nous ouvrons DataLab, et ouvrons la boîte de dialogue des paramètres (en utilisant « Fichier > Paramètres... », ou l’icône dans la barre d’outils).

### Voir aussi :

Le système de plugins est décrit dans la section [Plugins](#).

Ajoutons le plugin `cdd_example_imageproc.py` à DataLab (c’est un plugin d’exemple qui est livré avec le package source de DataLab, ou peut être téléchargé [ici sur GitHub](#)).

Si nous fermons et rouvrons DataLab, nous pouvons voir que le plugin est maintenant disponible dans le menu « Plugins » : il y a une nouvelle entrée « Extract blobs (exemple) ».

Pour information, l’image est générée par le plugin en utilisant le code suivant :

```
def generate_test_image(self) -> None:
    """Generate test image"""
    # Create a NumPy array:
    arr = np.random.normal(10000, 1000, (2048, 2048))
    for _ in range(10):
        row = np.random.randint(0, arr.shape[0])
        col = np.random.randint(0, arr.shape[1])
```

(suite sur la page suivante)

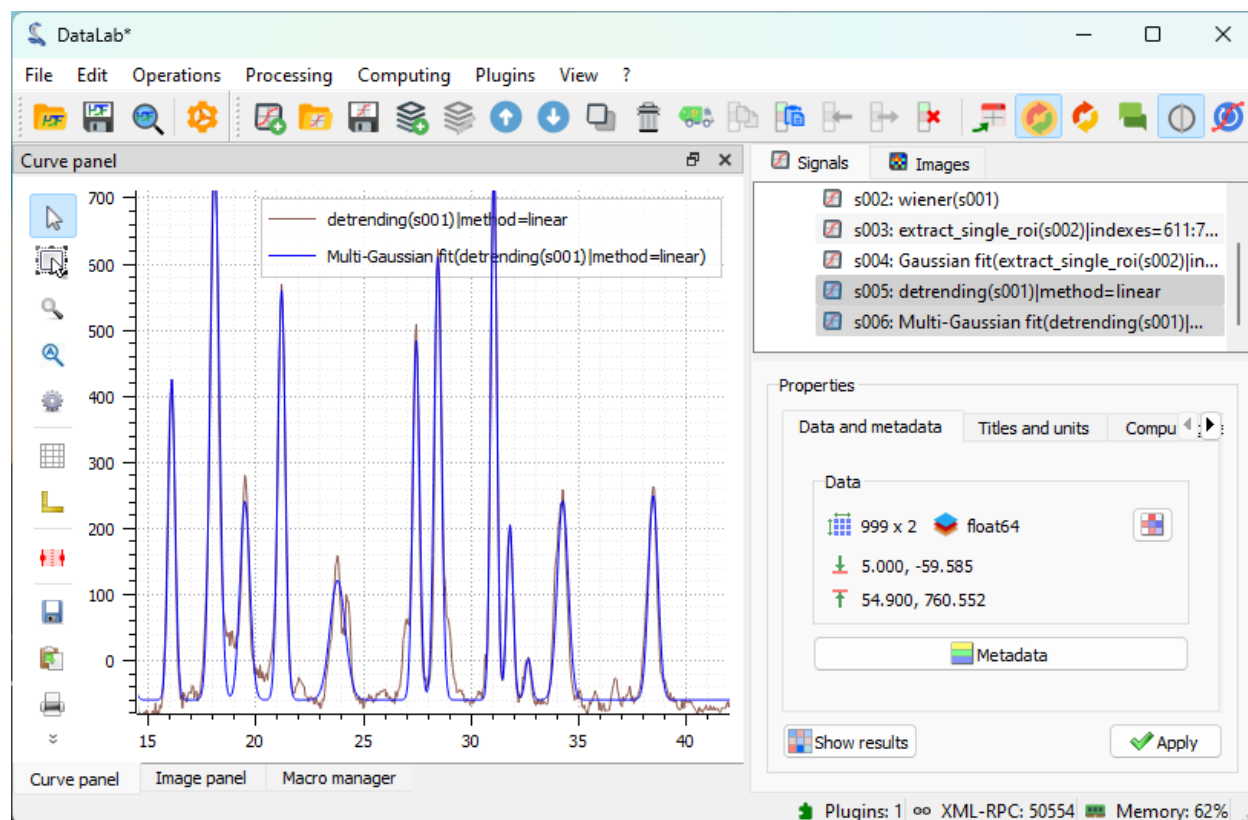


FIG. 22 – Le résultat de l’ajustement est affiché dans la fenêtre principale. Ici, nous avons sélectionné à la fois le spectre et l’ajustement dans le panneau « Signaux » à droite, donc les deux sont affichés dans le panneau de visualisation à gauche.

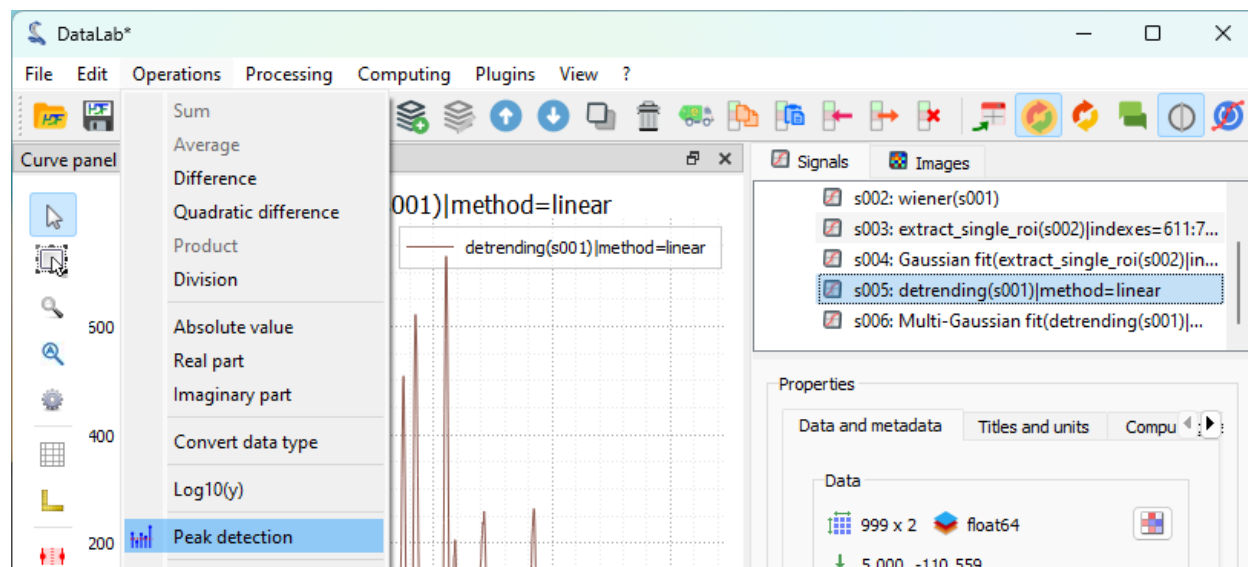


FIG. 23 – Ouvrir la fenêtre « Détection de pics » avec « Opérations > Détection de pics ».

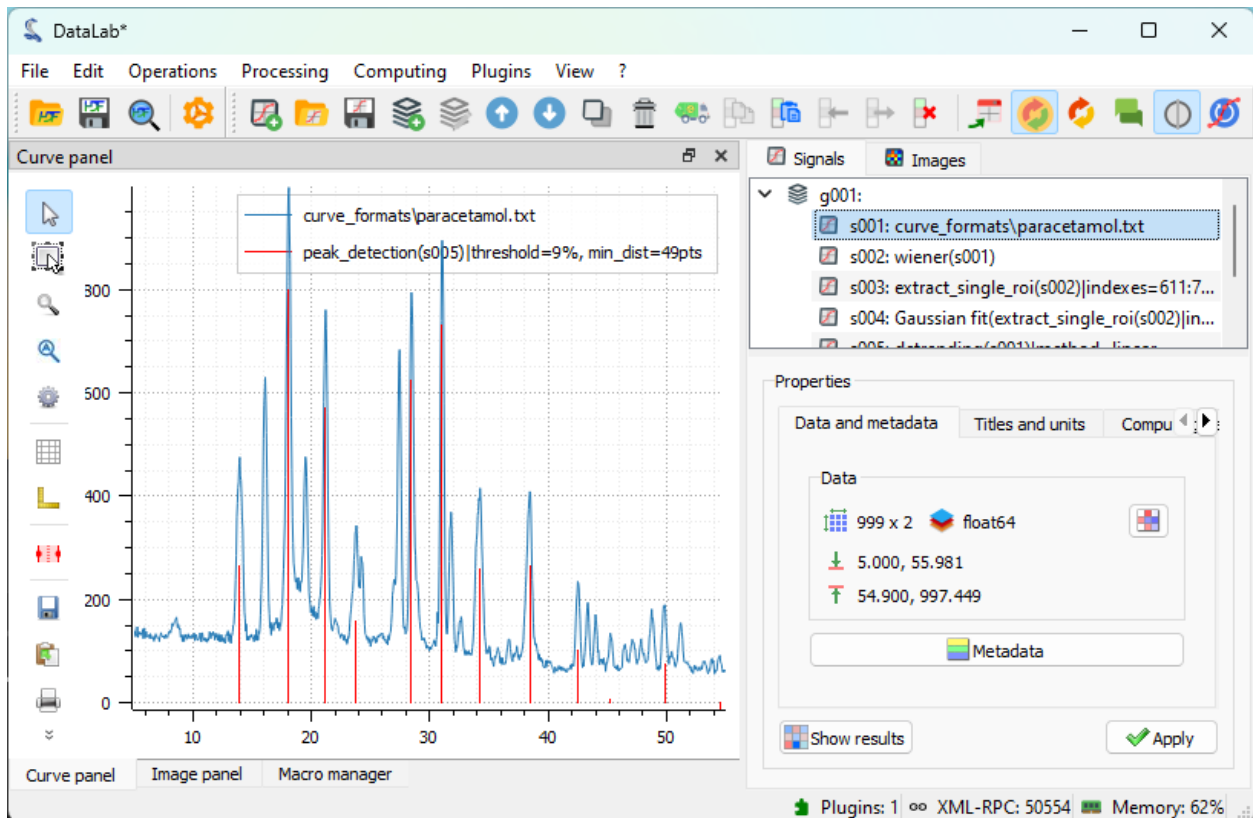


FIG. 24 – Après avoir ajusté les paramètres de la boîte de dialogue de détection de pics (même boîte de dialogue que celle utilisée pour l’ajustement multi-gaussien), cliquez sur « OK ». Ensuite, nous sélectionnons la « détection\_de\_pics » et le spectre d’origine dans le panneau « Signaux » à droite, de sorte que les deux soient affichés dans le panneau de visualisation à gauche.

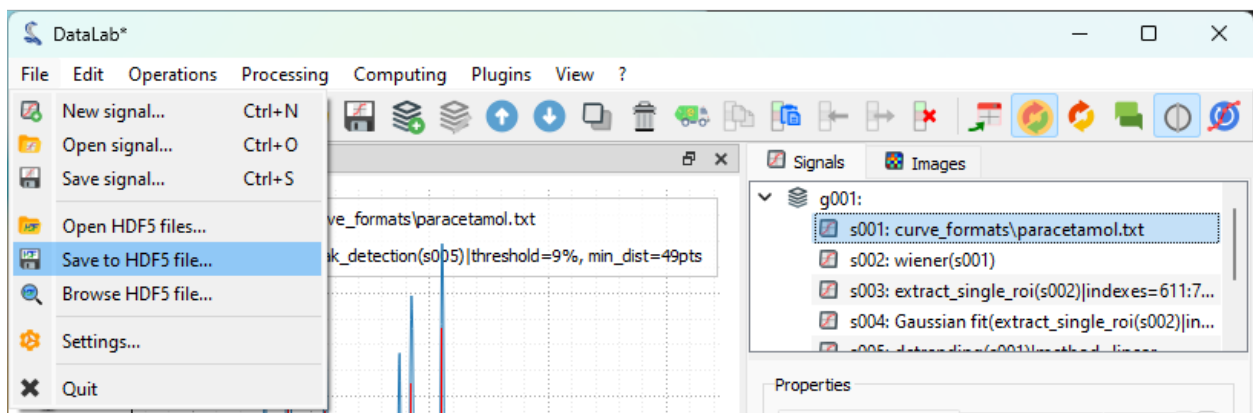


FIG. 25 – Sauvegarder l’espace de travail dans un fichier avec « Fichier > Sauvegarder dans un fichier HDF5... », ou le bouton dans la barre d’outils.

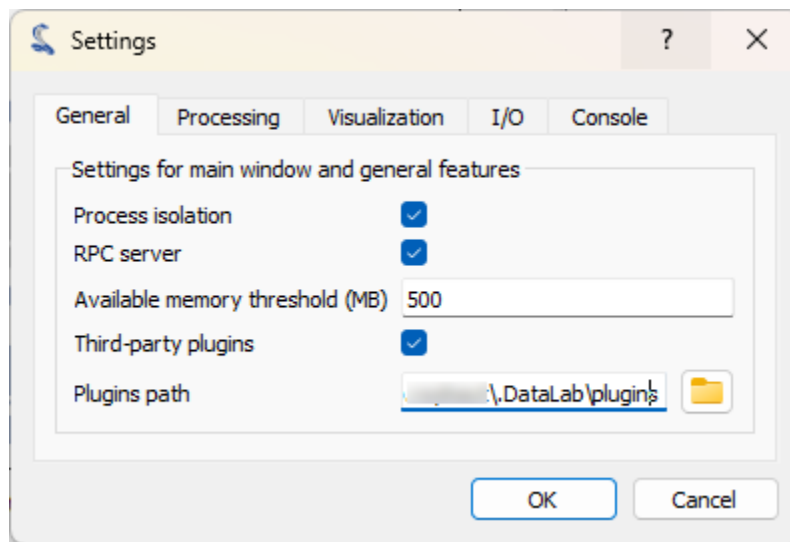


FIG. 26 – Dans l’onglet « Général », nous pouvons voir le champ « Chemin des plugins ». C’est le chemin où DataLab recherchera les plugins. Nous pouvons ajouter un nouveau plugin en copiant/collant le fichier du plugin dans ce répertoire.

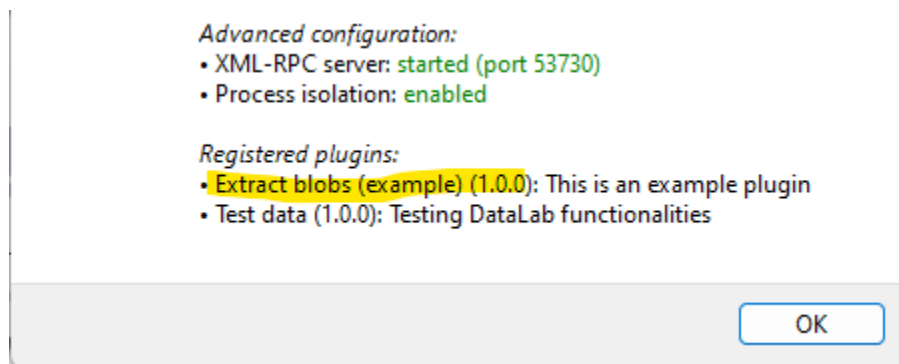


FIG. 27 – La boîte de dialogue « À propos de DataLab » affiche la liste des plugins disponibles.

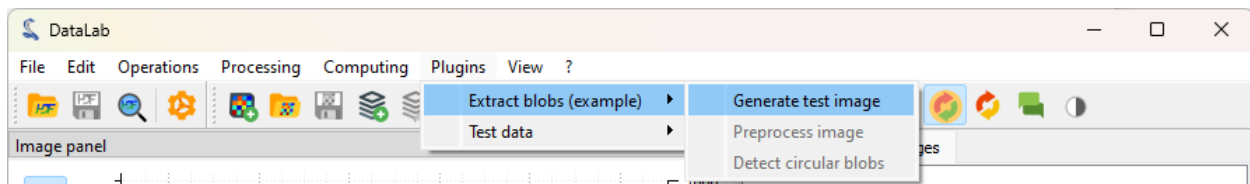


FIG. 28 – Cliquons sur « Extract blobs (example) > Generate test image »

(suite de la page précédente)

```

rr, cc = skimage.draw.disk((row, col), 40, shape=arr.shape)
arr[rr, cc] -= np.random.randint(5000, 6000)
icenter = arr.shape[0] // 2
rr, cc = skimage.draw.disk((icenter, icenter), 200, shape=arr.shape)
arr[rr, cc] -= np.random.randint(5000, 8000)
data = np.clip(arr, 0, 65535).astype(np.uint16)

# Create a new image object and add it to the image panel
image = cdl.obj.create_image("Test image", data, units=("mm", "mm", "lsb"))
self.proxy.add_object(image)

```

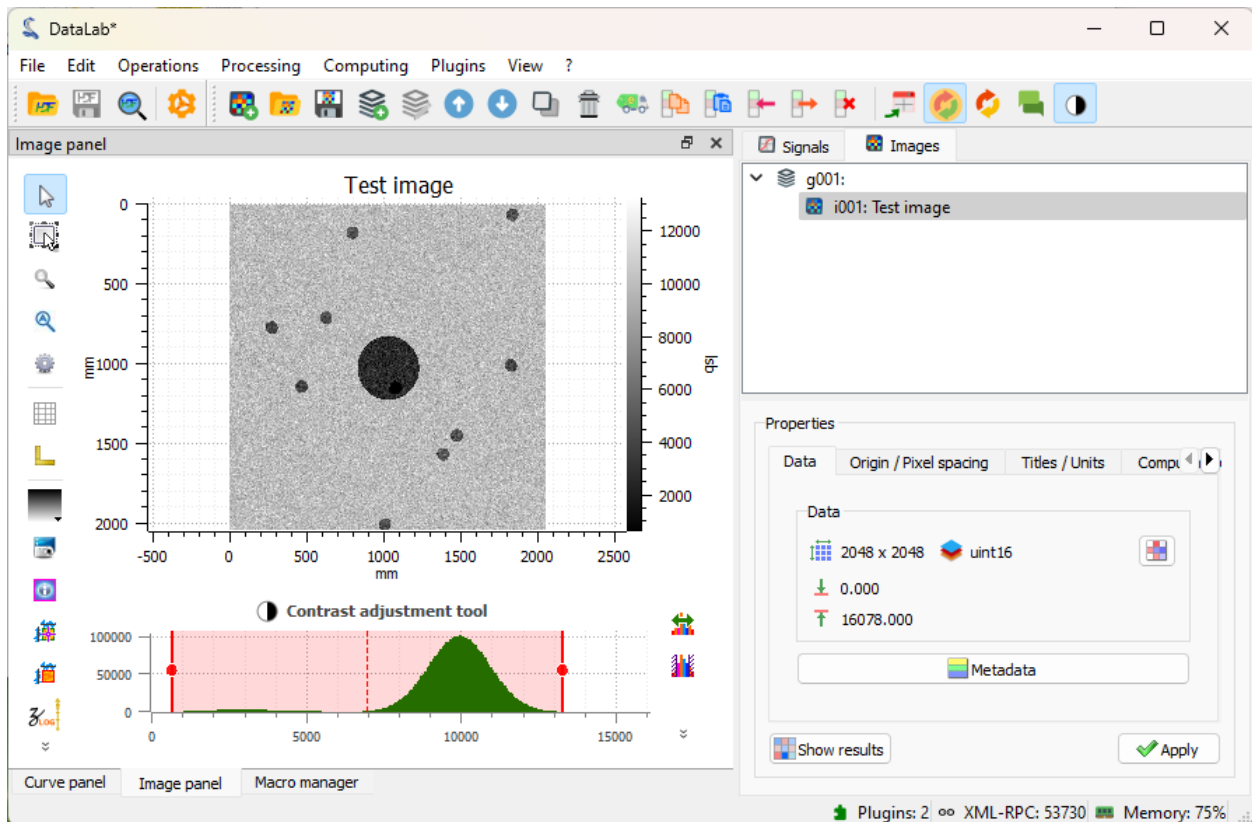


Fig. 29 – Le plugin a généré une image de test, et l’a ajoutée au panneau « Images ». L’image montre quelques taches, avec un disque sombre central, et un fond bruité.

Ce plugin a d’autres fonctionnalités, telles que le débruitage de l’image, et la détection de taches sur l’image, mais nous ne les couvrirons pas ici : nous utiliserons les mêmes fonctionnalités natives de DataLab que le plugin, manuellement.

L’image est un peu bruitée, et aussi assez grande. Réduisons la taille de l’image tout en la débruitant un peu en la binarisant par un facteur de 2.

Appliquons un filtre médian mobile à l’image, pour la débruiteur un peu plus.

A présent, détectons les taches sur l’image.

**Note :** Si vous souhaitez afficher à nouveau les résultats d’analyse, vous pouvez sélectionner l’entrée « Afficher les résultats » dans le menu Analyse », ou le bouton « Afficher les résultats » , en dessous de la liste des images :

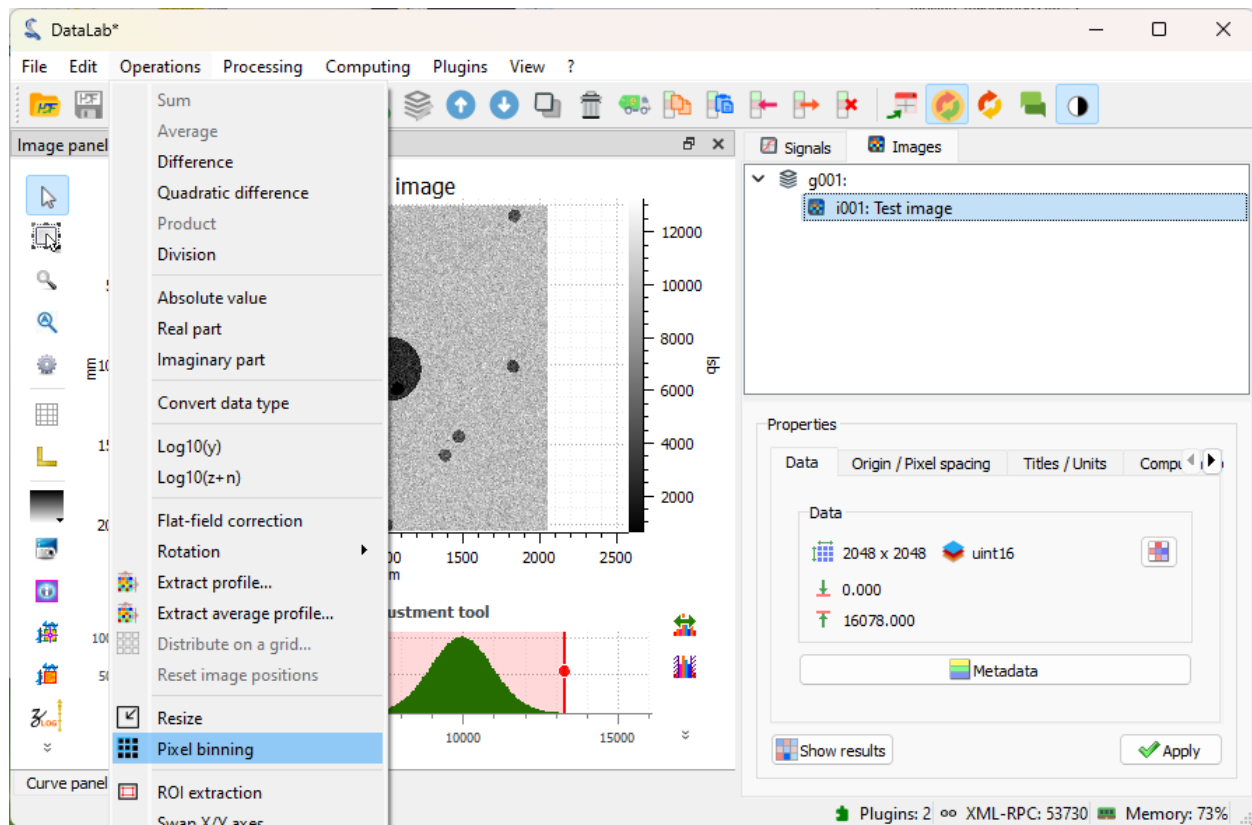


FIG. 30 – Cliques sur « Opérations > Pixel binning ».

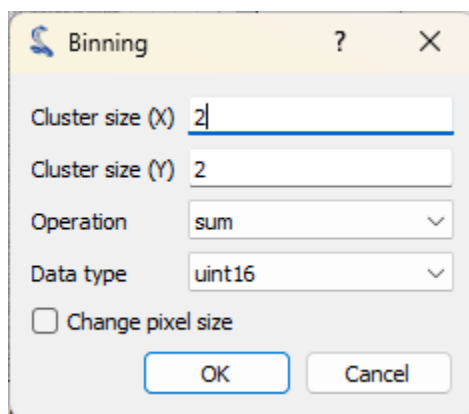


FIG. 31 – La boîte de dialogue « Binning » s'ouvre. Réglez le facteur de binning à 2, et cliquez sur « OK ».

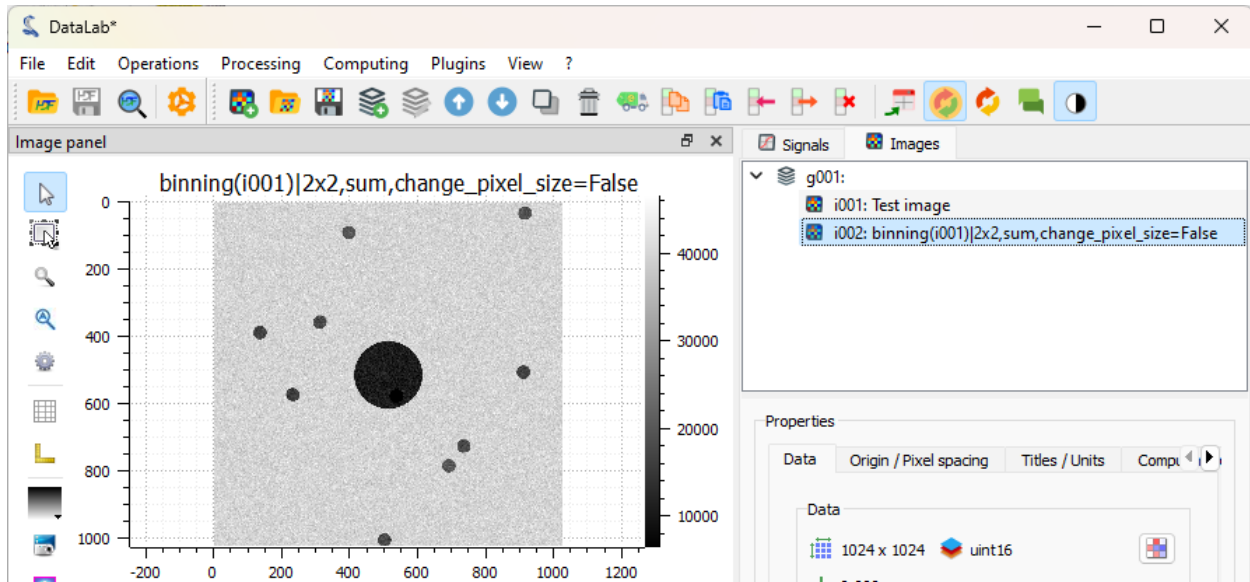


FIG. 32 – L'image binnée est ajoutée au panneau « Images ». Il est maintenant plus facile de voir les taches (même si elles étaient déjà assez visibles sur l'image d'origine : c'est juste un exemple), et l'image sera plus rapide à traiter.

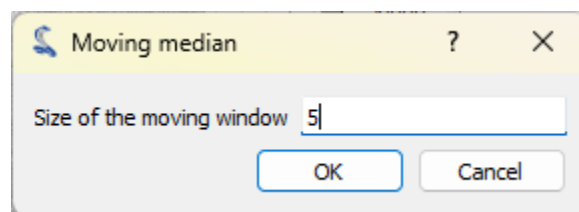


FIG. 33 – Cliquons sur l'entrée « Processing > Moving median », et réglez la taille de la fenêtre sur 5.

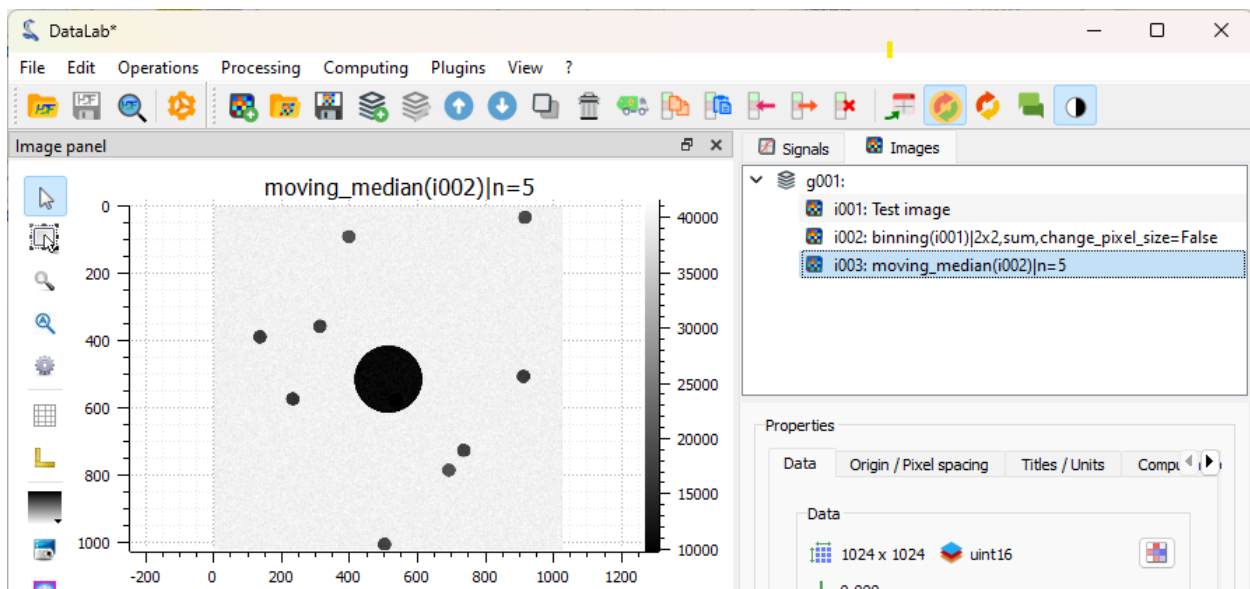


FIG. 34 – L'image filtrée est ajoutée au panneau « Images ». Le débruitage est assez efficace.

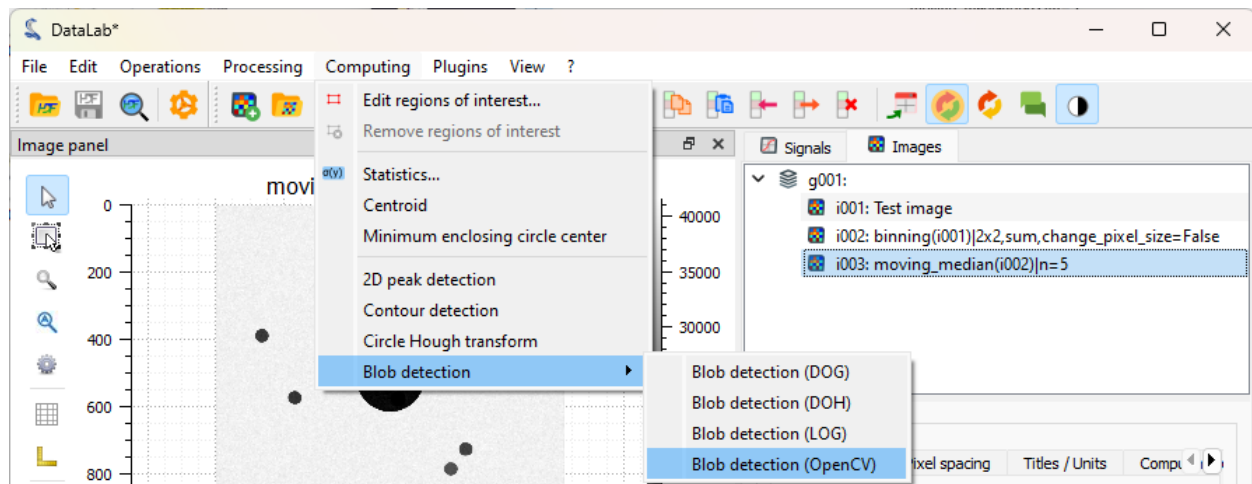


FIG. 35 – Cliquons sur « Analysis > Blob detection > Blob detection (OpenCV) ».

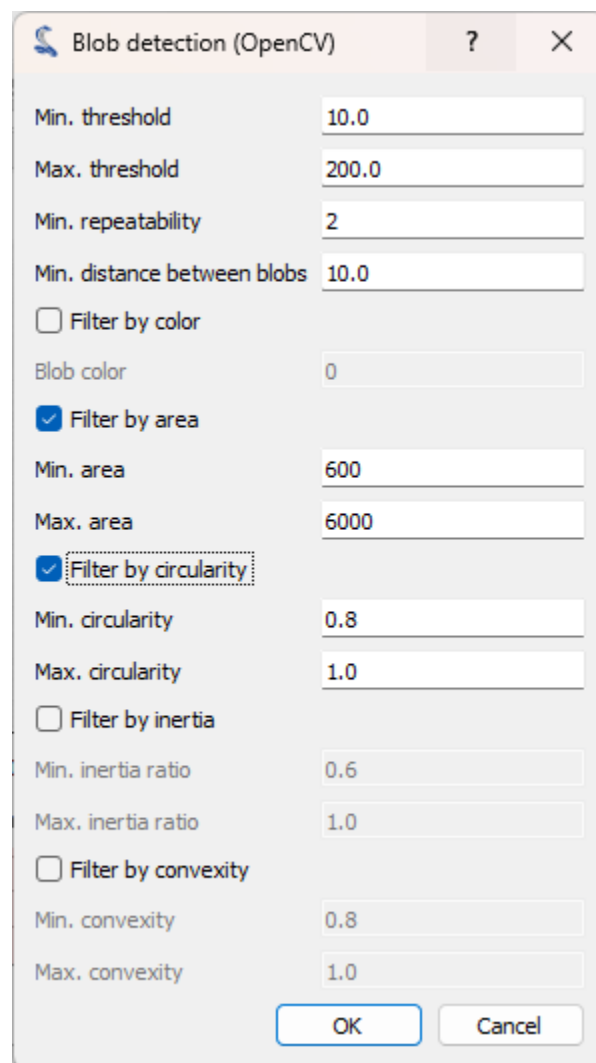
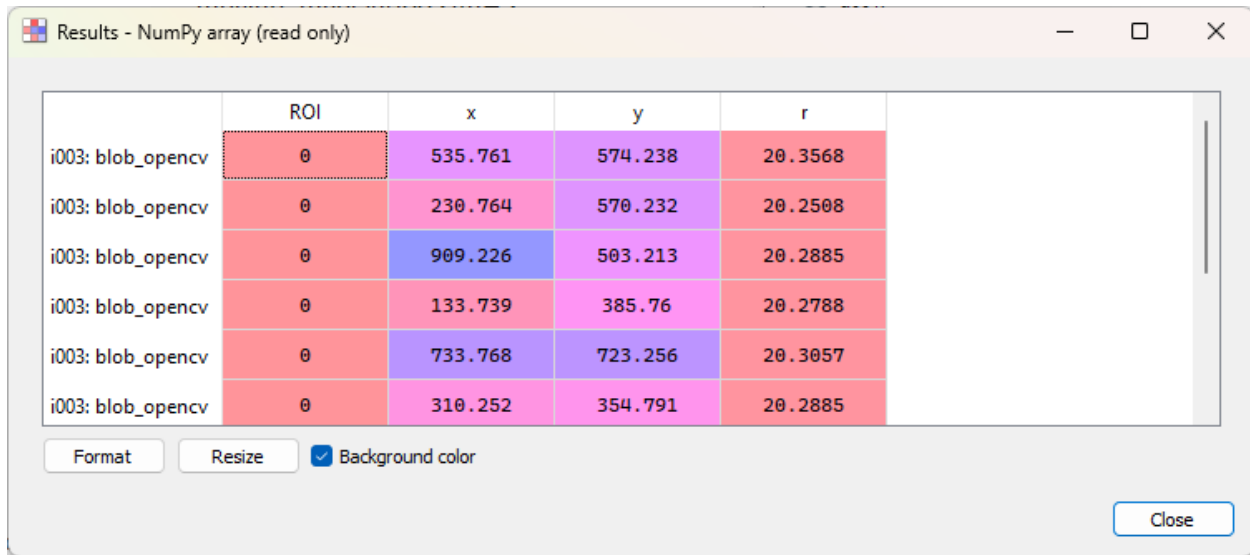
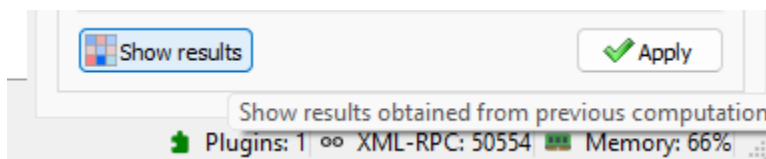


FIG. 36 – La boîte de dialogue « Blob detection (OpenCV) » s'ouvre. Réglez les paramètres comme indiqué sur la capture d'écran, et cliquez sur « OK ».



	ROI	x	y	r
i003: blob_opencv	0	535.761	574.238	20.3568
i003: blob_opencv	0	230.764	570.232	20.2508
i003: blob_opencv	0	909.226	503.213	20.2885
i003: blob_opencv	0	133.739	385.76	20.2788
i003: blob_opencv	0	733.768	723.256	20.3057
i003: blob_opencv	0	310.252	354.791	20.2885

Fig. 37 – La boîte de dialogue « Résultats » s’ouvre, montrant les taches détectées : une ligne par tache, avec les coordonnées et le rayon de la tache.



Enfin, nous pouvons enregistrer l’espace de travail dans un fichier. L’espace de travail contient toutes les images qui ont été chargées dans DataLab, ainsi que les résultats de traitement. Il contient également les paramètres de visualisation (palettes de couleurs, contraste, etc.), les métadonnées et les annotations. Pour enregistrer l’espace de travail, cliquez sur « Fichier > Enregistrer dans un fichier HDF5... », ou sur le bouton dans la barre d’outils.

Si vous souhaitez charger à nouveau l’espace de travail, vous pouvez utiliser « Fichier > Ouvrir un fichier HDF5... » (ou le bouton dans la barre d’outils) pour charger l’ensemble de l’espace de travail, ou « Fichier > Parcourir un fichier HDF5... » (ou le bouton dans la barre d’outils) pour charger uniquement une sélection d’ensembles de données de l’espace de travail.

## Mesure de franges de Fabry-Perot

Cet exemple montre comment mesurer des franges de Fabry-Perot à l’aide des fonctionnalités de traitement d’image de DataLab :

- Charger une image d’un interféromètre de Fabry-Perot
- Définir une région d’intérêt circulaire (ROI) autour de la frange centrale
- Détecter les contours dans la ROI et les ajuster à des cercles
- Afficher le rayon des cercles
- Annoter l’image
- Copier/coller la ROI dans une autre image
- Extraire le profil d’intensité le long de l’axe X
- Sauvegarder l’espace de travail

Tout d’abord, nous ouvrons DataLab et chargeons les images :

L’image sélectionnée est affichée dans la fenêtre principale. Nous pouvons zoomer en appuyant sur le bouton droit de la souris et en faisant glisser la souris vers le haut et vers le bas. Nous pouvons également déplacer l’image en appuyant

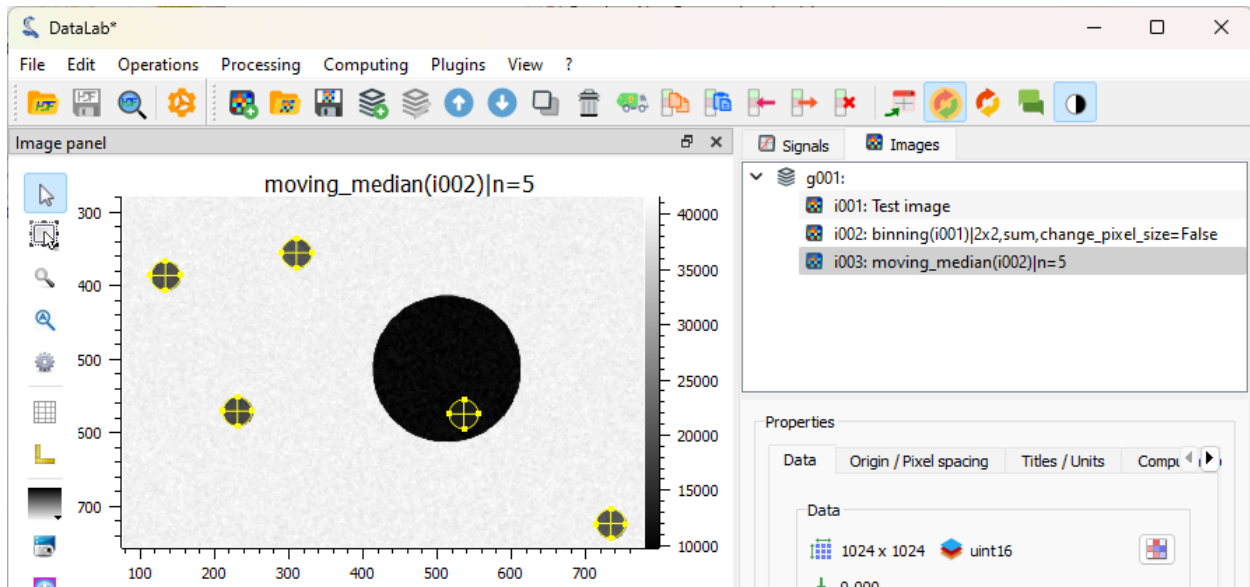


FIG. 38 – Les taches détectées sont également ajoutées aux métadonnées de l'image, et peuvent être vues dans le panneau de visualisation à gauche.

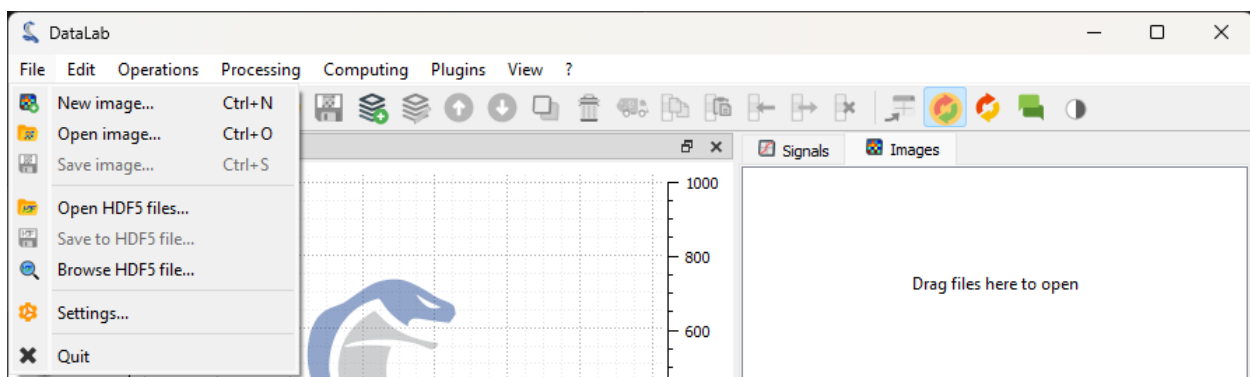


FIG. 39 – Ouvrir les fichiers d'image avec « Fichier > Ouvrir... », ou avec le bouton dans la barre d'outils, ou en faisant glisser-déposer les fichiers dans DataLab (sur le panneau de droite).

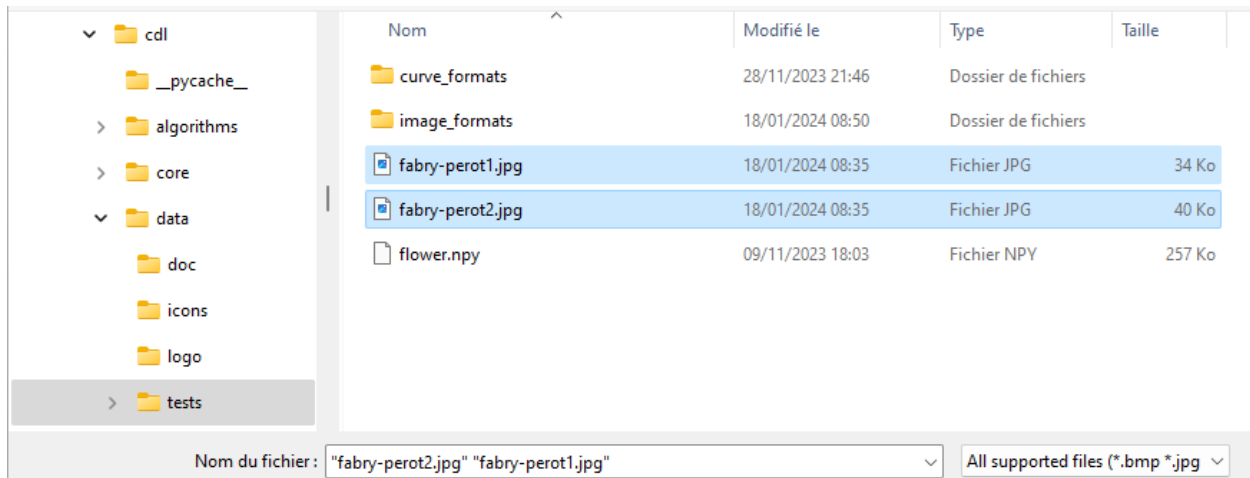


FIG. 40 – Sélectionnez les images de test « fabry\_perot1.jpg » et « fabry\_perot2.jpg » et cliquez sur « Ouvrir ».

sur le bouton du milieu de la souris et en faisant glisser la souris.

**Note :** Lorsque vous travaillez sur des images spécifiques à une application (par exemple des images de radiographie X, ou des images de microscopie optique), il est souvent utile de changer la palette de couleurs en une palette de couleurs en niveaux de gris. Si vous voyez une palette de couleurs d'image différente de celle affichée dans l'image, vous pouvez la modifier en sélectionnant l'image dans le panneau de visualisation, puis en sélectionnant la palette de couleurs dans la barre d'outils verticale à gauche du panneau de visualisation.

Ou, encore mieux, vous pouvez modifier la palette de couleurs par défaut dans les paramètres de DataLab en sélectionnant « Edition > Paramètres... » dans le menu, ou le bouton dans la barre d'outils.

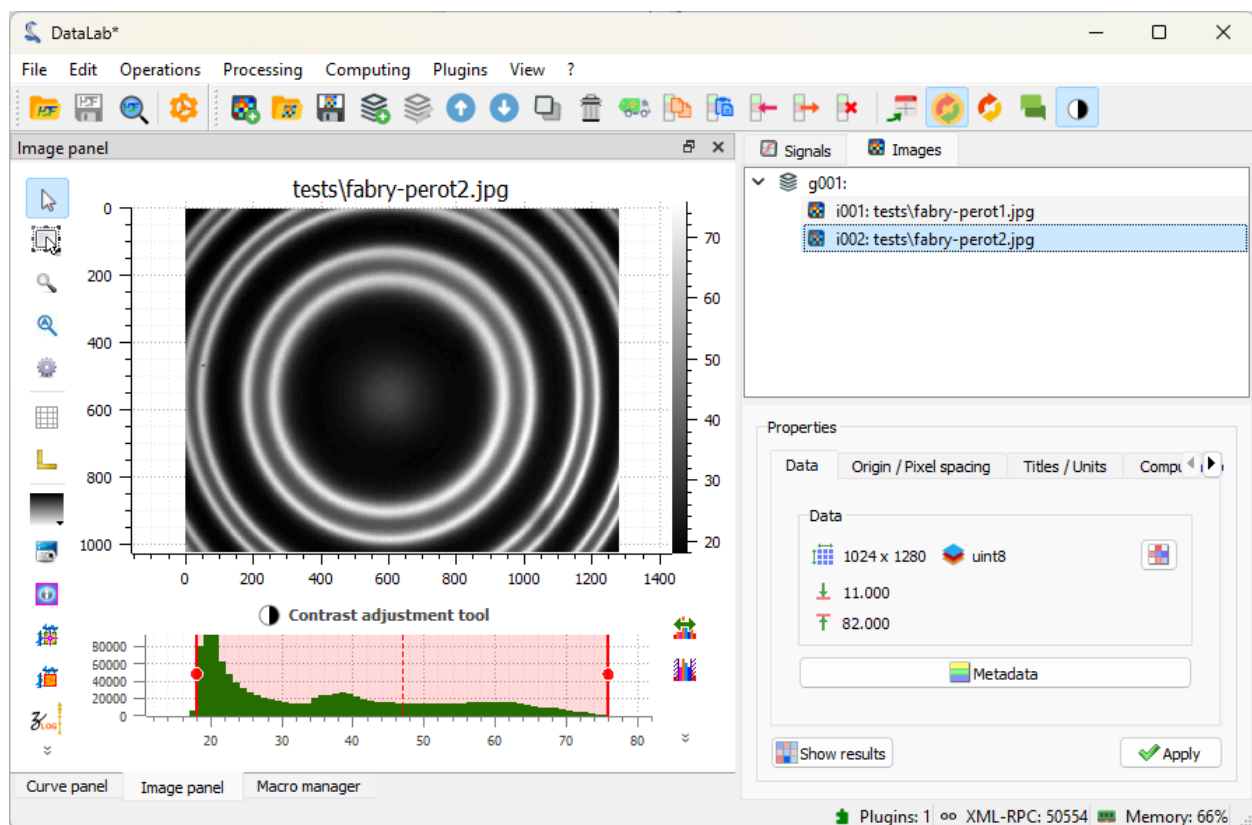


FIG. 41 – Zoomer avec le bouton droit de la souris. Déplacer l'image avec le bouton du milieu de la souris.

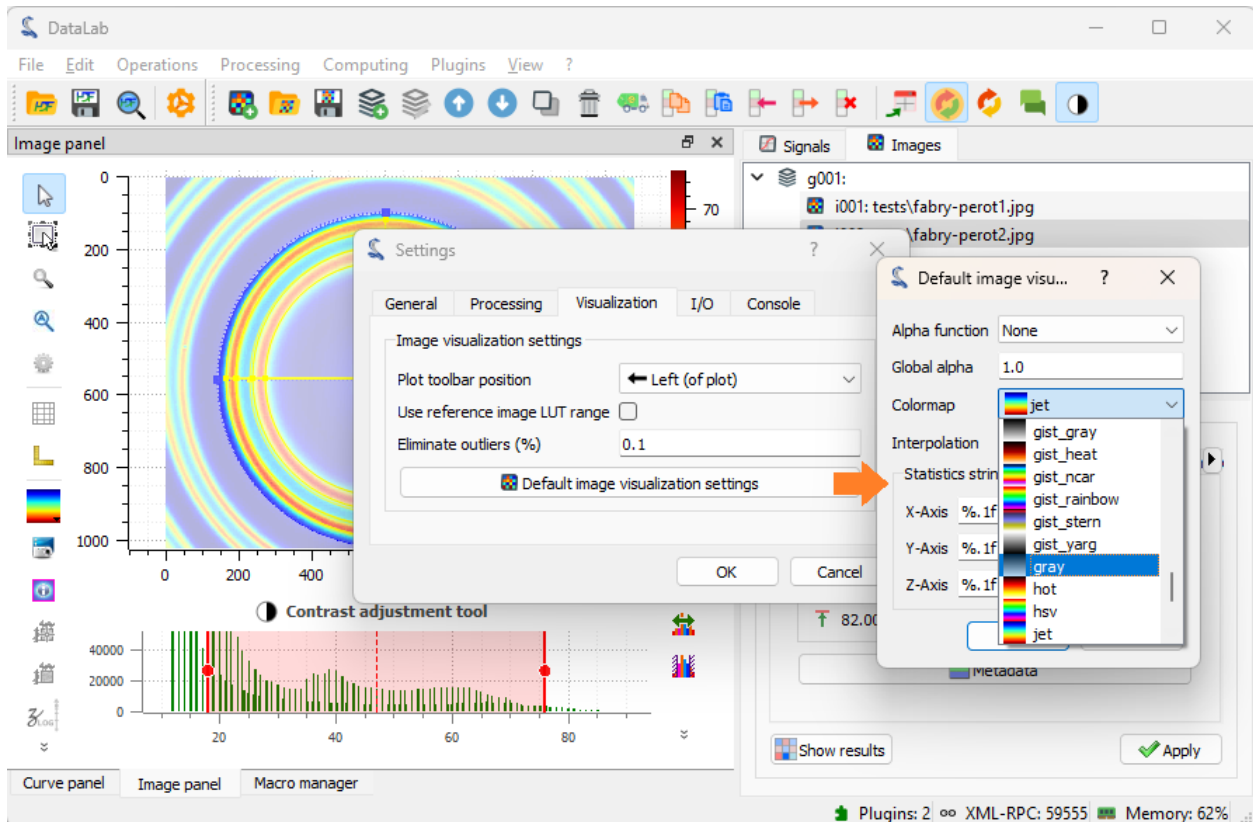


FIG. 42 – Sélectionnez l'onglet « Visualisation », et sélectionnez la palette de couleurs « gray ».

Ensuite, définissons une région d'intérêt circulaire (ROI) autour de la frange centrale.

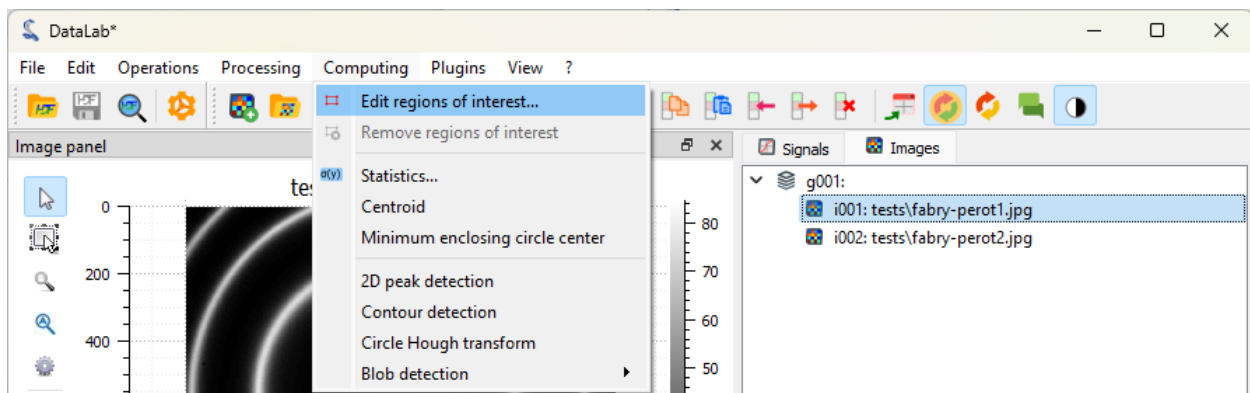


FIG. 43 – Sélectionnez l'outil « Modifier les régions d'intérêt » dans le menu Analyse ».

A présent, détectons les contours dans la ROI et ajustons-les à des cercles.

**Note :** Si vous souhaitez afficher à nouveau les résultats d'analyse, vous pouvez sélectionner l'entrée « Afficher les résultats » dans le menu Analyse », ou le bouton « Afficher les résultats », en dessous de la liste des images :

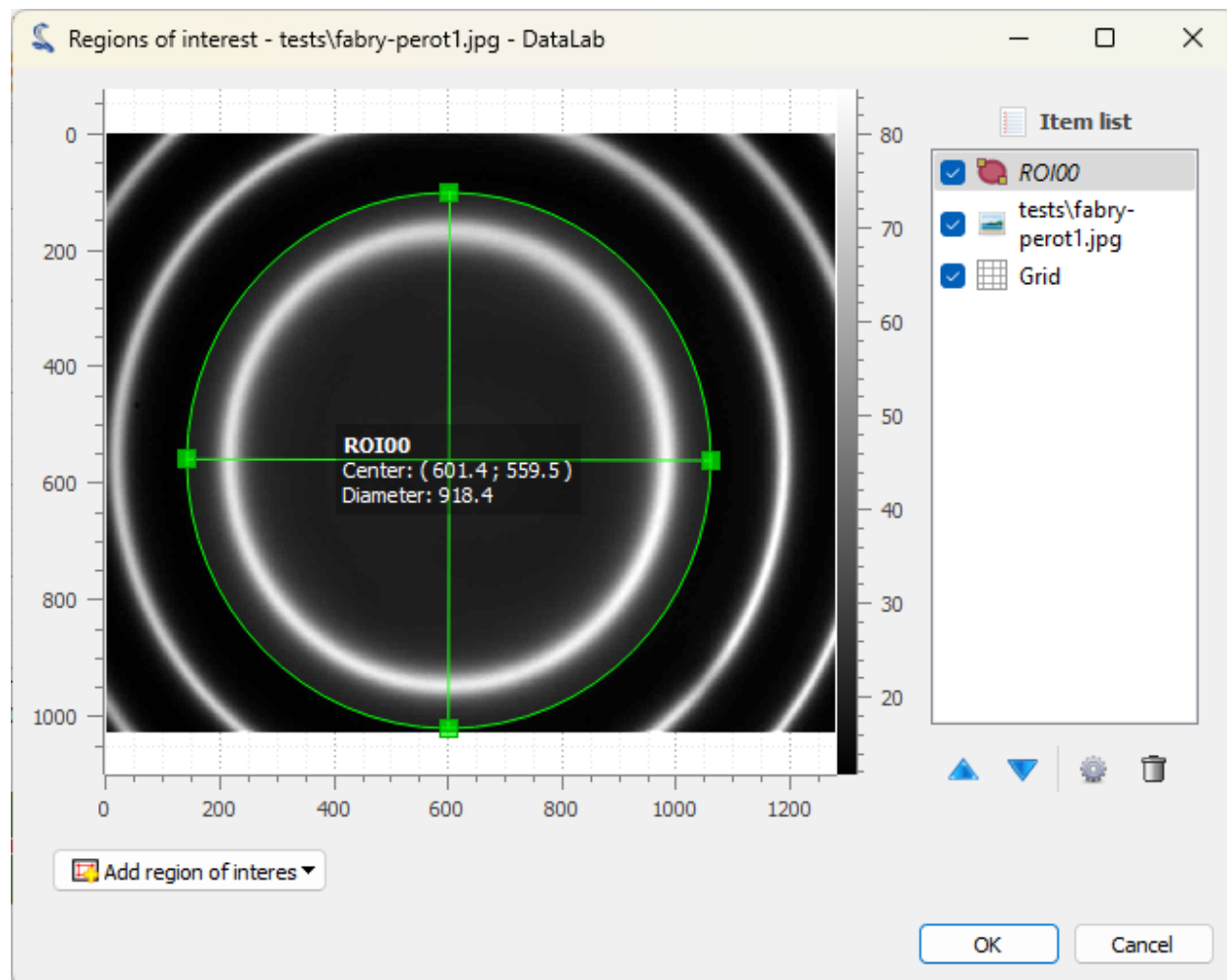


FIG. 44 – La boîte de dialogue « Régions d'intérêt » s'ouvre. Cliquez sur « Ajouter une ROI » et sélectionnez une ROI circulaire. Redimensionnez la ROI prédéfinie en faisant glisser les poignées. Notez que vous pouvez modifier le rayon de la ROI tout en gardant son centre fixe en appuyant sur la touche « Ctrl ». Cliquez sur « OK » pour fermer la boîte de dialogue.

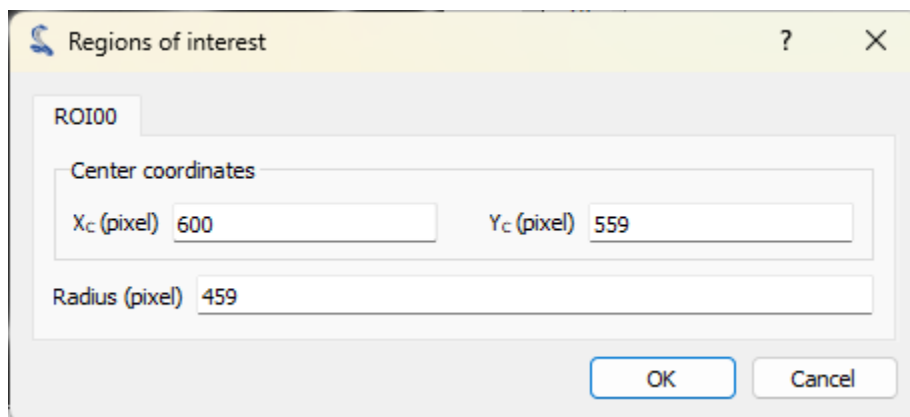


FIG. 45 – Une autre boîte de dialogue s'ouvre et vous demande de confirmer les paramètres de la ROI. Cliquez sur « OK ».

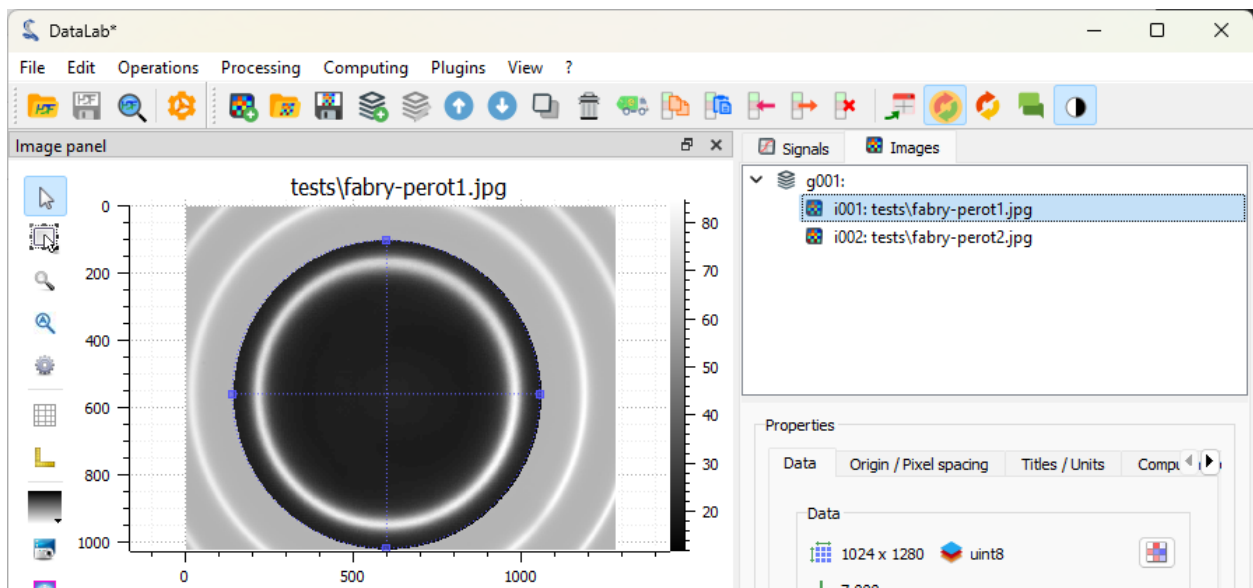


FIG. 46 – La ROI est affichée sur l'image : les pixels masqués sont grisés, et la limite de la ROI est affichée en bleu (notez que, en interne, la ROI est définie par un masque binaire, c'est-à-dire que les données d'image sont représentées sous la forme d'un tableau masqué NumPy).

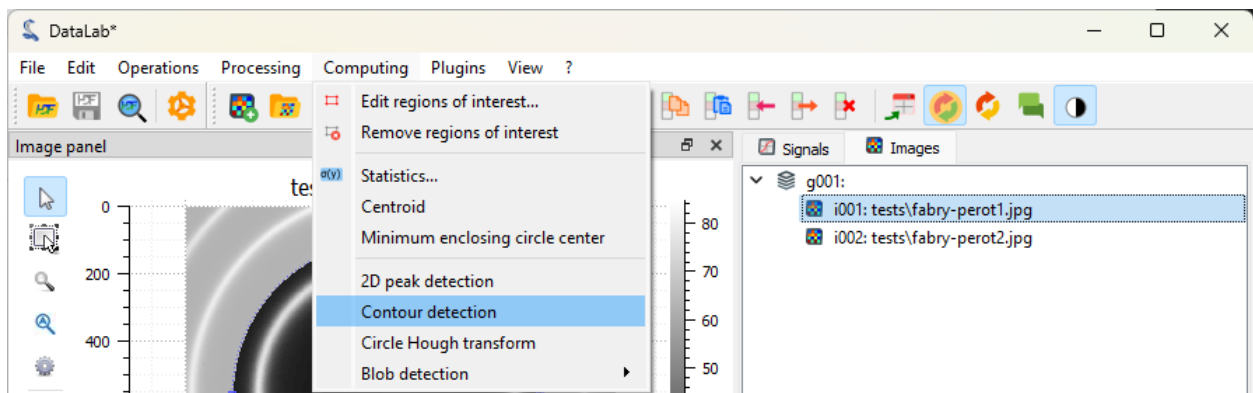


FIG. 47 – Sélectionnez l'outil « Détection de contour » dans le menu Analyse ».

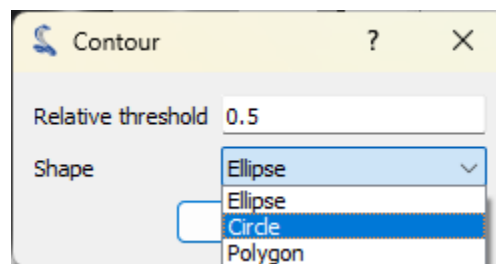


FIG. 48 – La boîte de dialogue « Contour » s'ouvre. Sélectionnez la forme « Cercle » et cliquez sur « OK ».

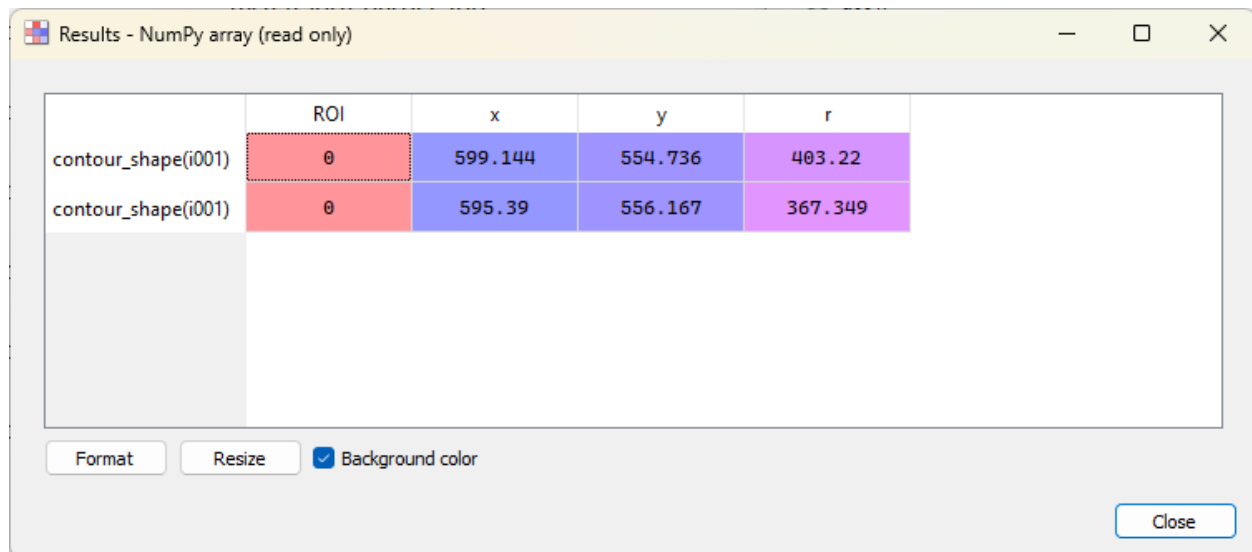


FIG. 49 – La boîte de dialogue « Résultats » s'ouvre et affiche les paramètres du cercle ajusté. Cliquez sur « OK ».

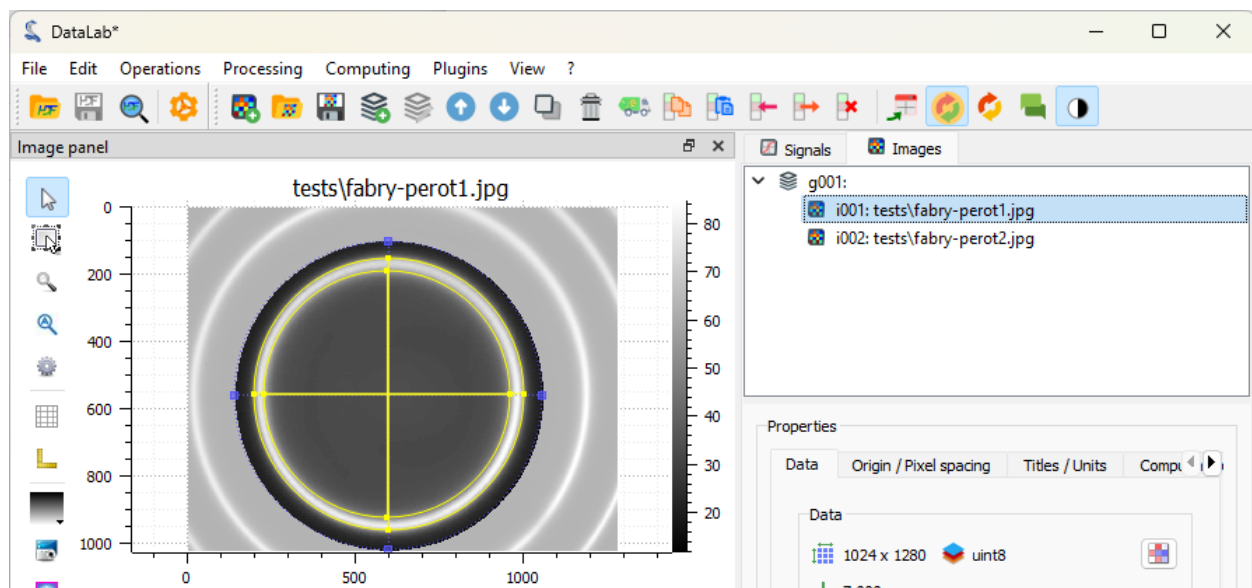
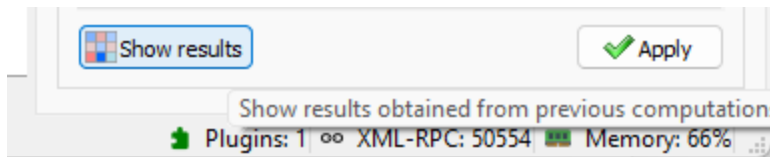


FIG. 50 – Les cercles ajustés sont affichés sur l'image.



Les images (ou signaux) peuvent également être affichés dans une fenêtre séparée, en cliquant sur l'entrée « Afficher dans une nouvelle fenêtre » dans le menu « Affichage » (ou le bouton dans la barre d'outils). Ceci est utile pour comparer côte à côte des images ou des signaux.

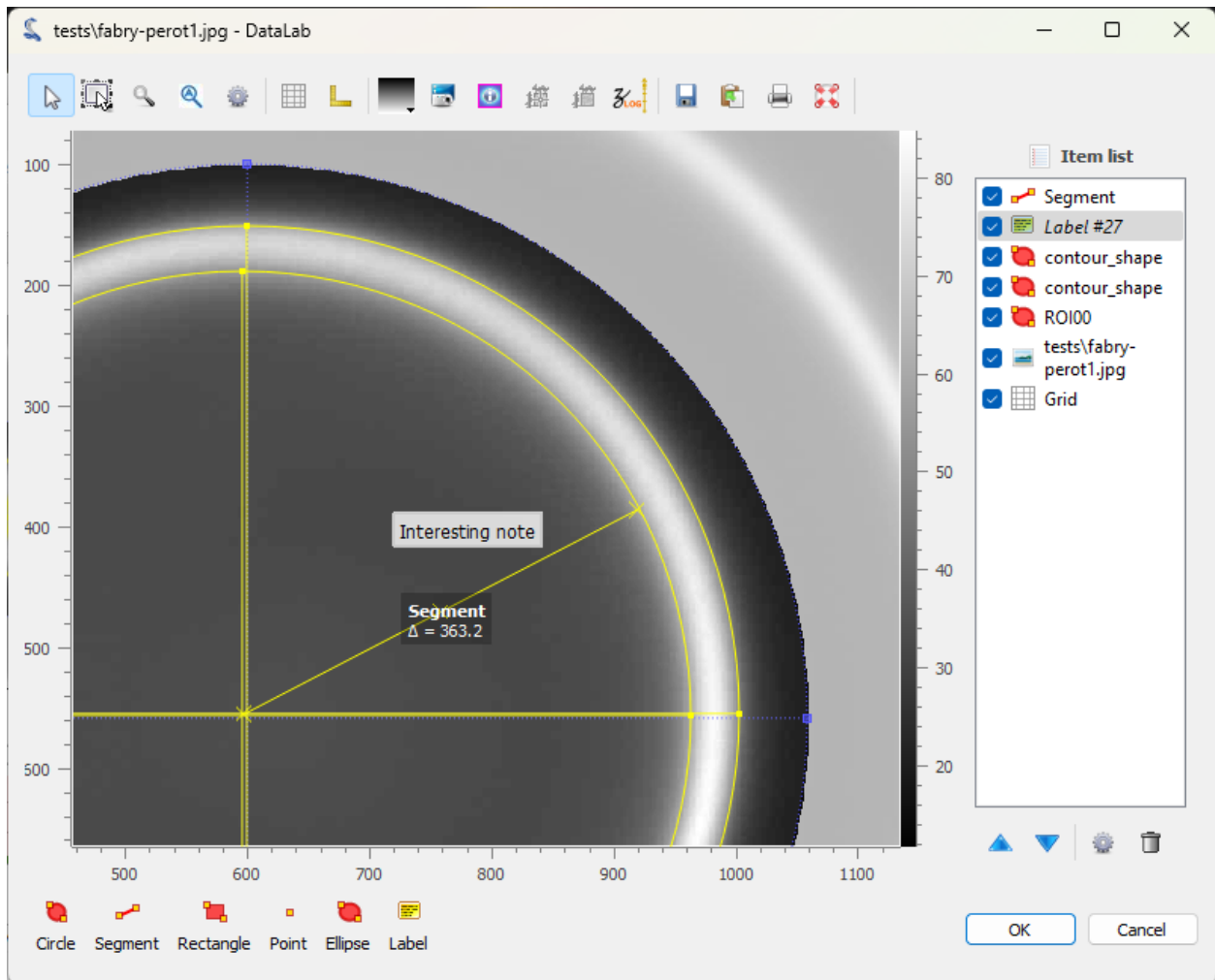


FIG. 51 – L'image est affichée dans une fenêtre séparée. La ROI et les cercles ajustés sont également affichés. Des annotations peuvent être ajoutées à l'image en cliquant sur les boutons en bas de la fenêtre. Les annotations sont stockées dans les métadonnées de l'image, et avec les données de l'image lorsque l'espace de travail est sauvegardé. Cliquez sur « OK » pour fermer la fenêtre.

Si vous souhaitez examiner de plus près les métadonnées, vous pouvez ouvrir la boîte de dialogue « Métadonnées ».

Maintenant, supprimons les métadonnées de l'image (y compris les annotations) pour nettoyer l'image.

Si nous voulons définir la même ROI sur la deuxième image, nous pouvons copier/coller la ROI de la première image vers la deuxième image, en utilisant les métadonnées.

Pour extraire le profil d'intensité le long de l'axe X, nous avons deux options :

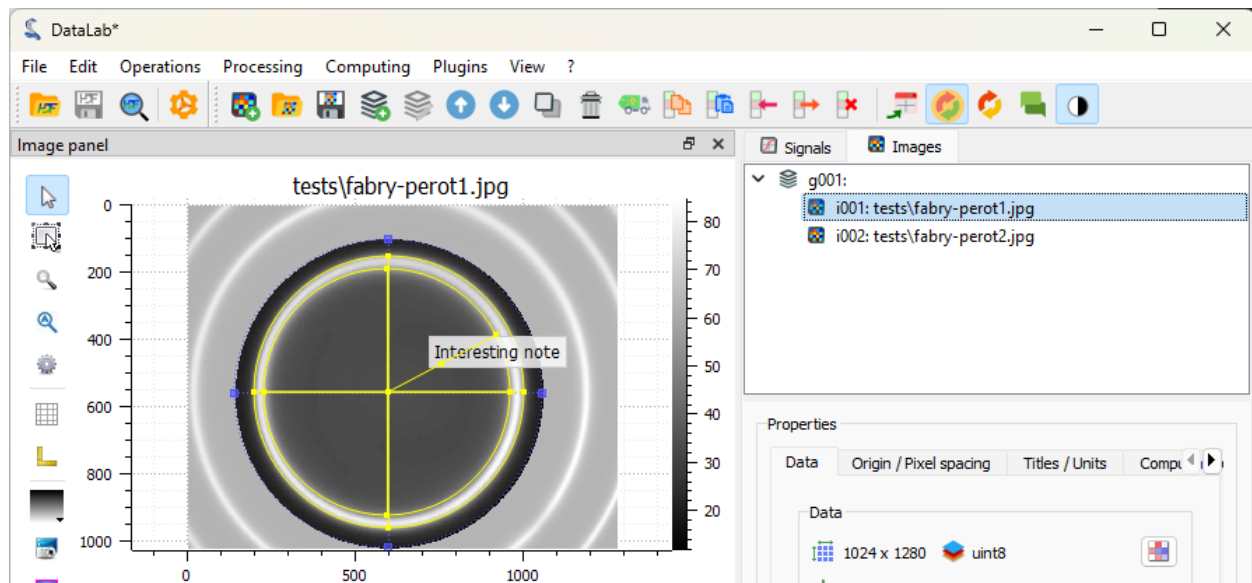


FIG. 52 – L'image est affichée dans la fenêtre principale, avec les annotations.

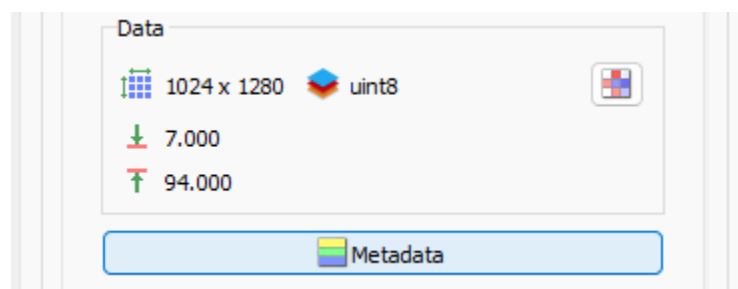


FIG. 53 – Le bouton « Métadonnées » est situé en dessous de la liste des images.

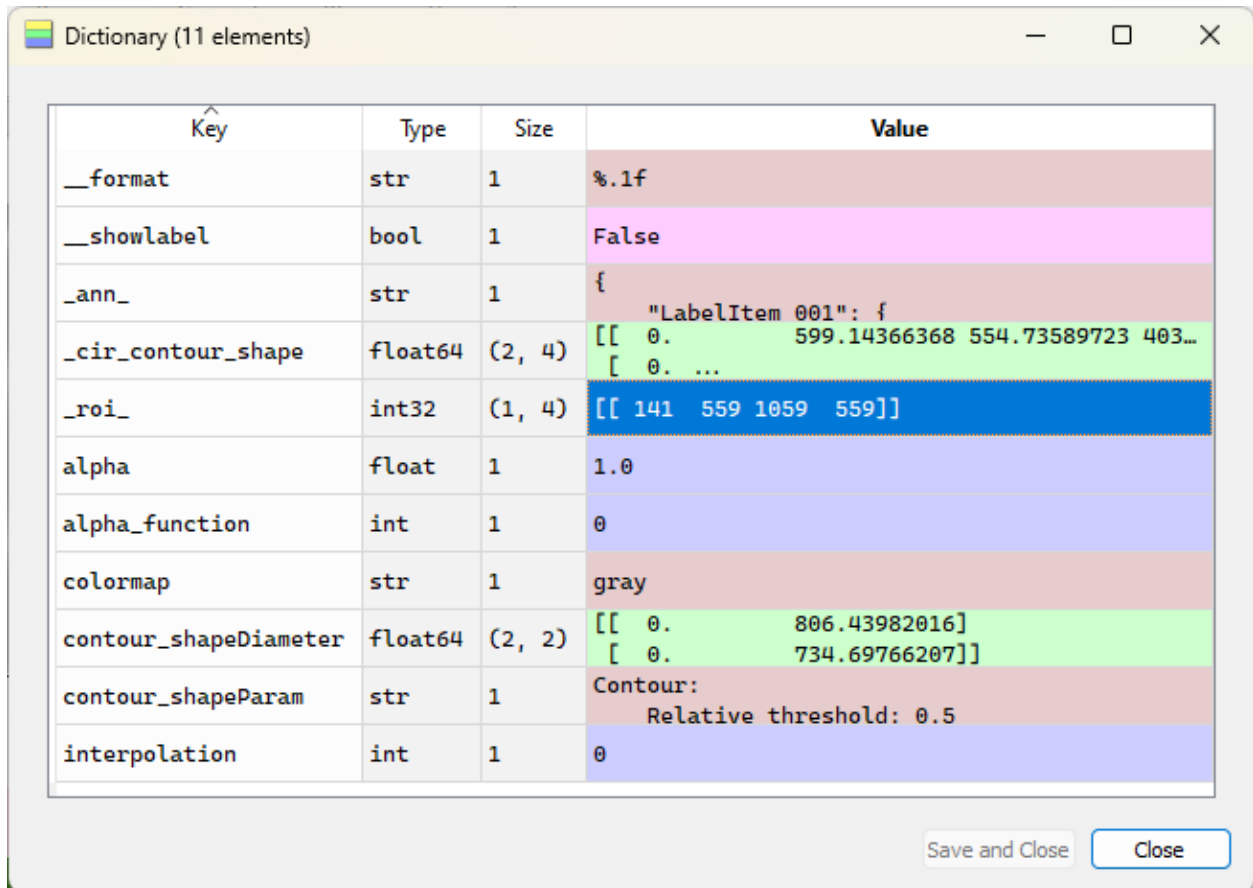


FIG. 54 – La boîte de dialogue « Métadonnées » s'ouvre. Parmi d'autres informations, elle affiche les annotations (dans un format JSON), des informations de style (par exemple la palette de couleurs), et la ROI.

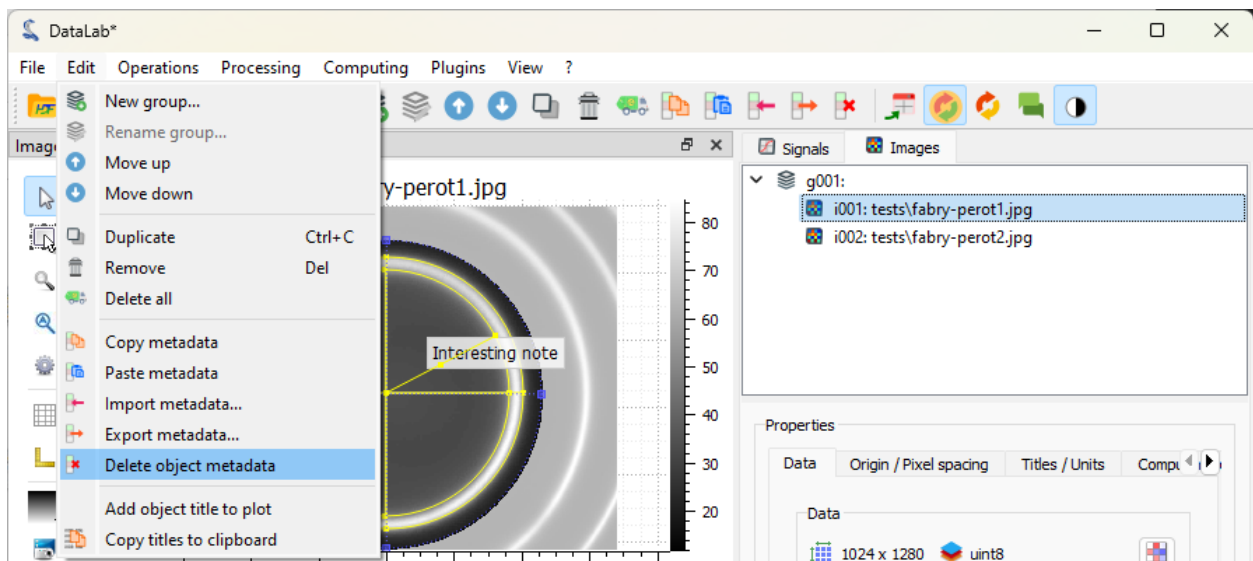


FIG. 55 – Sélectionnez l'entrée « Supprimer les métadonnées » dans le menu « Edition », ou le bouton dans la barre d'outils.

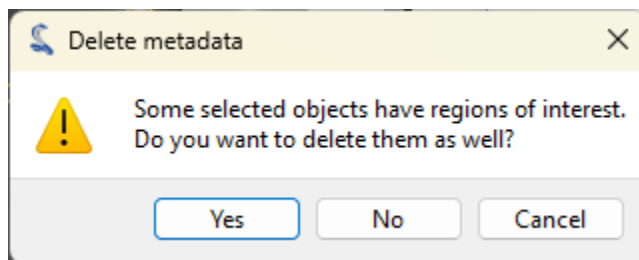


Fig. 56 – La boîte de dialogue « Supprimer les métadonnées » s'ouvre. Cliquez sur « Non » pour conserver la ROI et supprimer le reste des métadonnées.

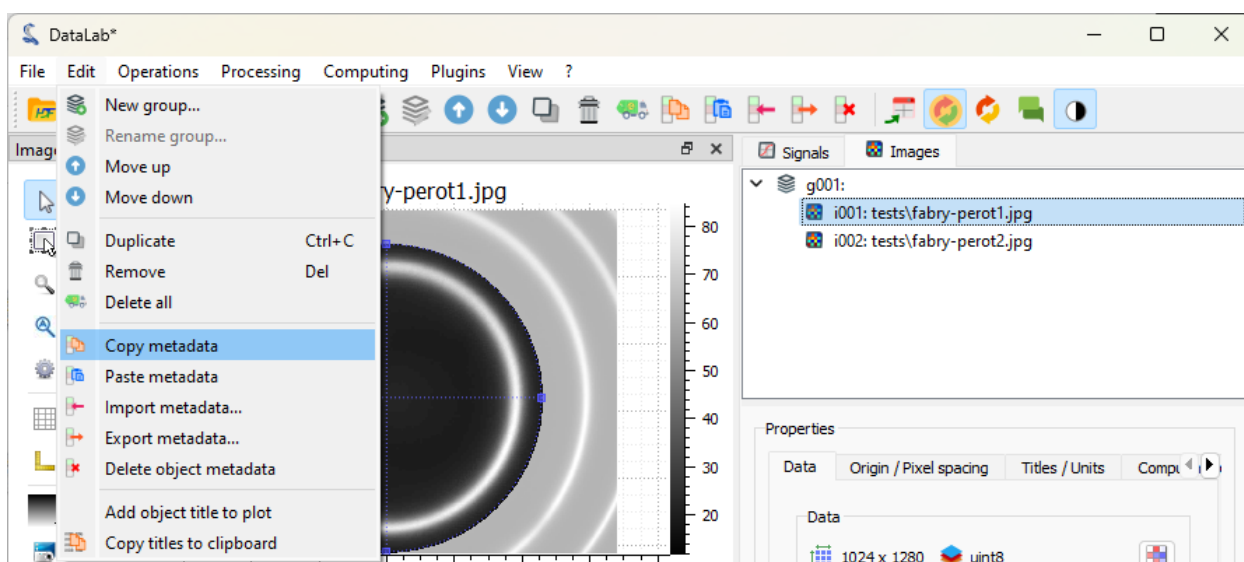


Fig. 57 – Sélectionnez l'entrée « Copier les métadonnées » dans le menu « Edition », ou le bouton dans la barre d'outils.

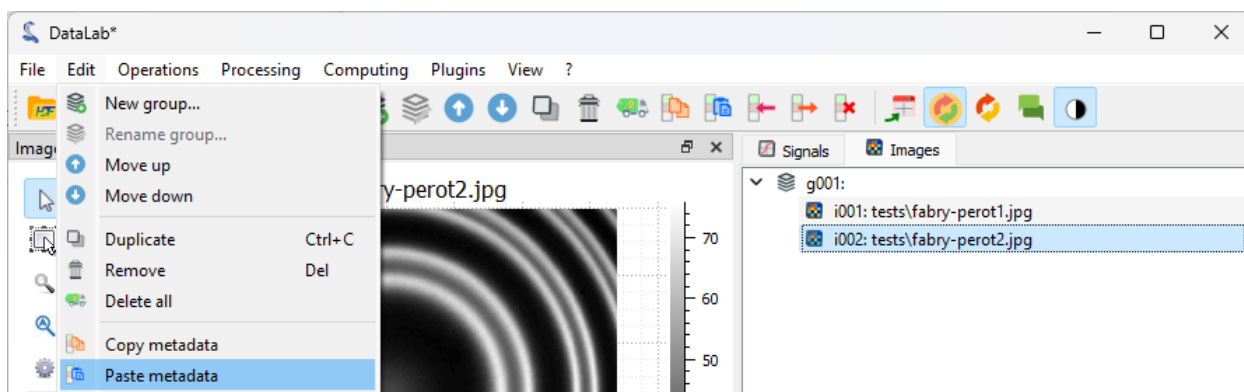


Fig. 58 – Sélectionnez la deuxième image dans le panneau « Images », puis sélectionnez l'entrée « Coller les métadonnées » dans le menu « Edition », ou le bouton dans la barre d'outils.

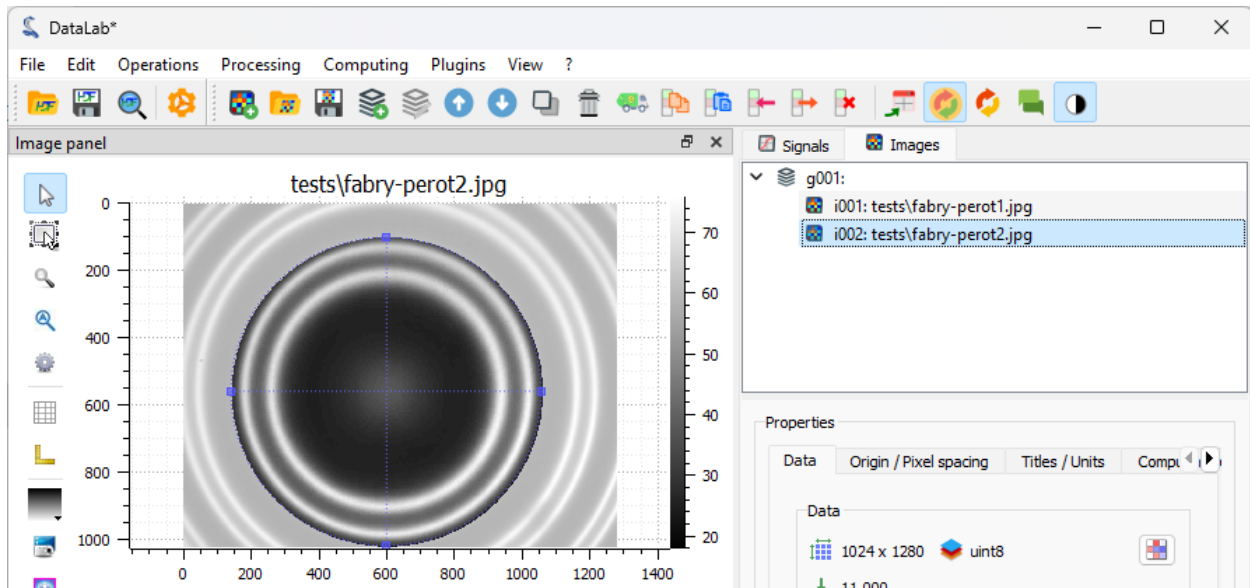


Fig. 59 – La ROI est ajoutée à la deuxième image.



Fig. 60 – Sélectionnez l'outil « Détection de contour » dans le menu Analyse », avec les mêmes paramètres qu'auparavant (forme « Cercle »). Sur cette image, il y a deux franges, donc quatre cercles sont ajustés. La boîte de dialogue « Résultats » s'ouvre et affiche les paramètres du cercle ajusté. Cliquez sur « OK ».

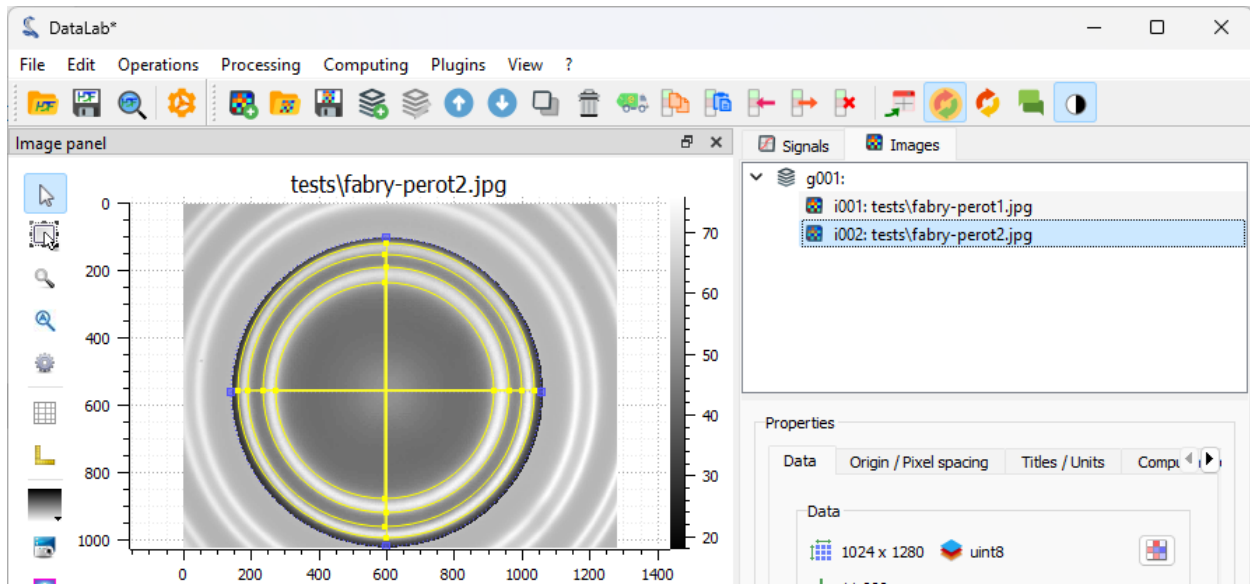


FIG. 61 – Les cercles ajustés sont affichés sur l'image.

- Soit sélectionner l'entrée « Profil rectiligne... » dans le menu « Opérations > Profils d'intensité ».
- Soit activer l'outil « Profil rectiligne » dans la barre d'outils verticale à gauche du panneau de visualisation. Essayons la première option, en sélectionnant l'entrée « Extraire le profil... » : c'est la façon la plus simple d'extraire un profil d'une image, et cela correspond à la méthode `compute_profile` de l'API de DataLab (donc elle peut être utilisée dans un script, un plugin ou une macro).

Si vous souhaitez effectuer des mesures sur le profil, ou ajouter des annotations, vous pouvez ouvrir le signal dans une fenêtre séparée, en cliquant sur l'entrée « Afficher dans une nouvelle fenêtre » dans le menu « Affichage » (ou le bouton dans la barre d'outils).

Maintenant, essayons la deuxième option pour extraire le profil d'intensité le long de l'axe X, en activant l'outil « Profil rectiligne » dans la barre d'outils verticale à gauche du panneau de visualisation (cet outil est une fonctionnalité de [PlotPy](#)). Avant de pouvoir l'utiliser, nous devons sélectionner l'image dans le panneau de visualisation (sinon l'outil est grisé). Ensuite, nous pouvons cliquer sur l'image pour afficher le profil d'intensité le long de l'axe X. DataLab intègre une version modifiée de cet outil, qui permet de transférer le profil vers le panneau « Signaux » pour un traitement ultérieur.

Ensuite, cliquez sur le bouton « Traiter le signal » dans la barre d'outils près du profil pour transférer le profil vers le panneau « Signaux ».

Enfin, nous pouvons sauvegarder l'espace de travail dans un fichier. L'espace de travail contient toutes les images et signaux qui ont été chargés ou traités dans DataLab. Il contient également les résultats d'analyse, les paramètres de visualisation (colormaps, contraste, etc.), les métadonnées et les annotations.

Si vous souhaitez charger à nouveau l'espace de travail, vous pouvez utiliser « Fichier > Ouvrir un fichier HDF5... » (ou le bouton dans la barre d'outils) pour charger l'ensemble de l'espace de travail, ou « Fichier > Parcourir un fichier HDF5... » (ou le bouton dans la barre d'outils) pour charger uniquement une sélection d'ensembles de données de l'espace de travail.

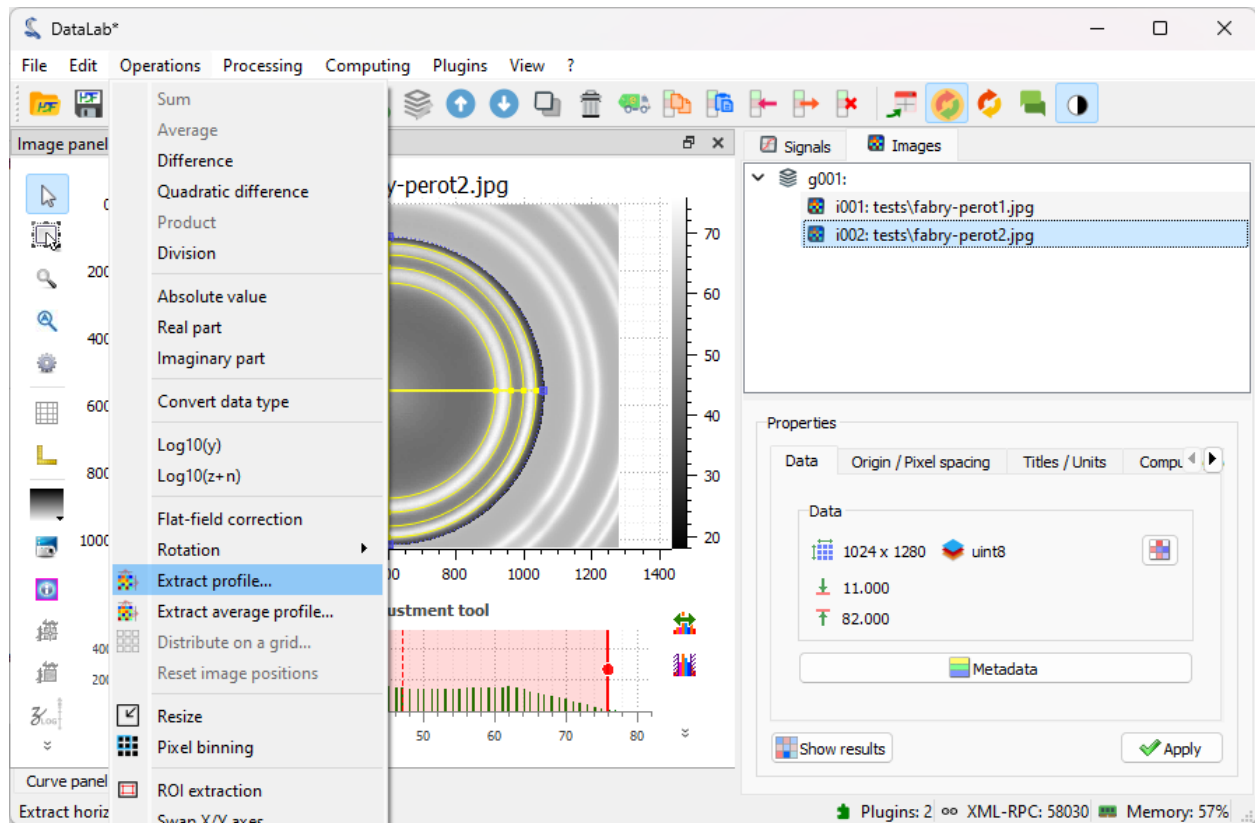


FIG. 62 – Sélectionnez l'entrée « Profil rectiligne... » dans le menu « Opérations ».

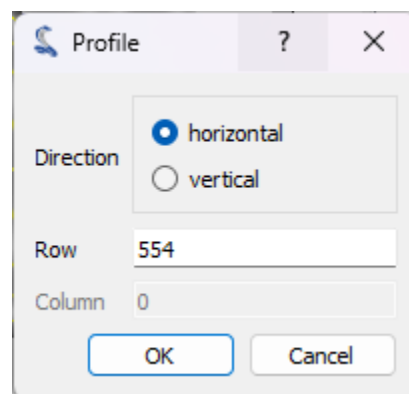


FIG. 63 – La boîte de dialogue « Profil » s'ouvre. Entrez la ligne du profil horizontal (ou la colonne du profil vertical) dans la boîte de dialogue qui s'ouvre. Cliquez sur « OK ».

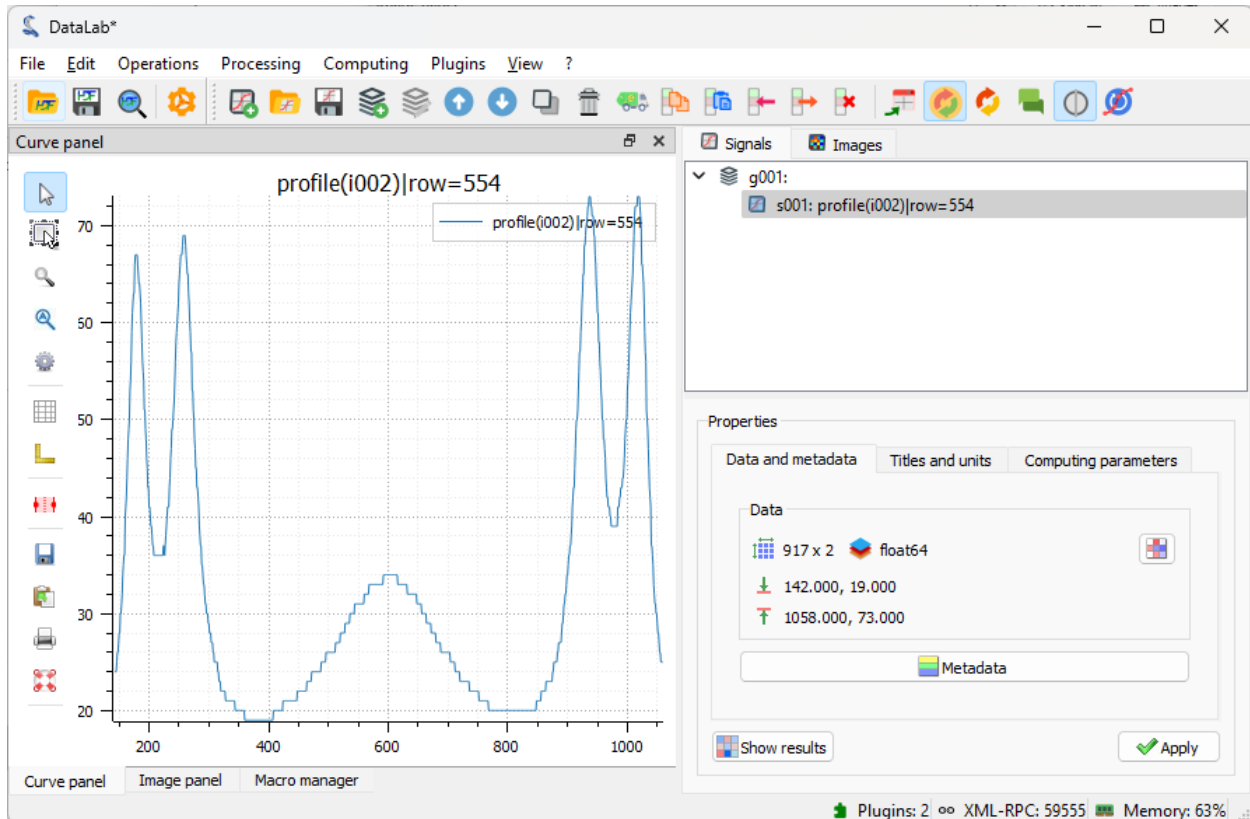


FIG. 64 – Le profil d'intensité est ajouté au panneau « Signaux », et DataLab bascule vers ce panneau pour afficher le profil.

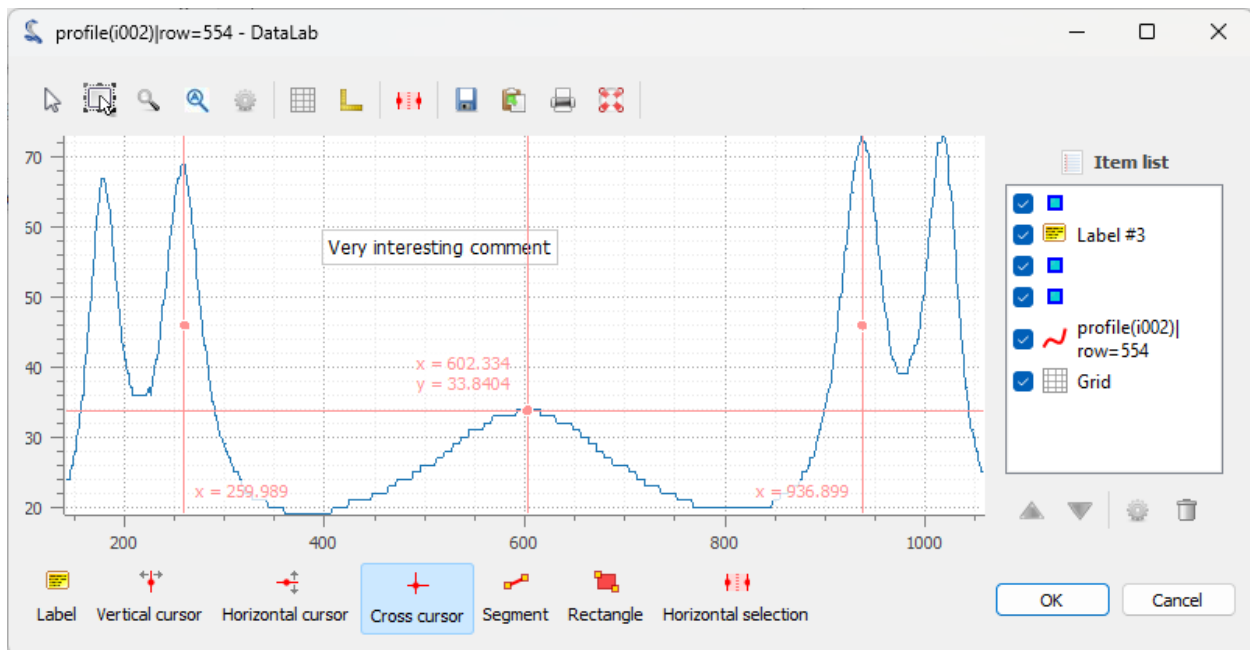


FIG. 65 – Le signal est affiché dans une fenêtre séparée. Ici, nous avons ajouté des curseurs verticales et une étiquette de texte très intéressante. Comme pour les images, les annotations sont stockées dans les métadonnées du signal, et avec les données du signal lorsque l'espace de travail est sauvegardé. Cliquez sur « OK » pour fermer la fenêtre.

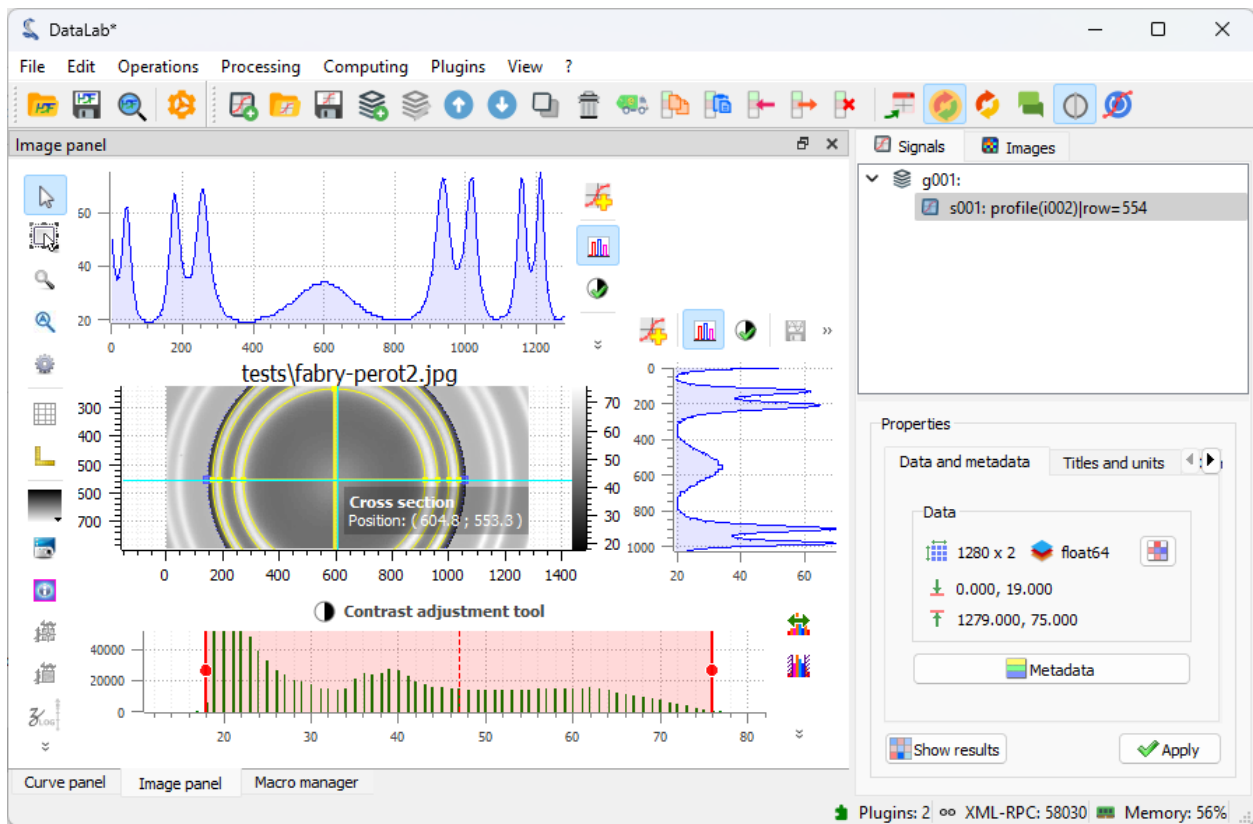


FIG. 66 – Revenez au panneau « Images » et sélectionnez l'image dans le panneau de visualisation (sinon l'outil « Section transversale » est grisée). Sélectionnez l'outil « Section transversale » dans la barre d'outils verticale, et cliquez sur l'image pour afficher les profils d'intensité le long des axes X et Y.

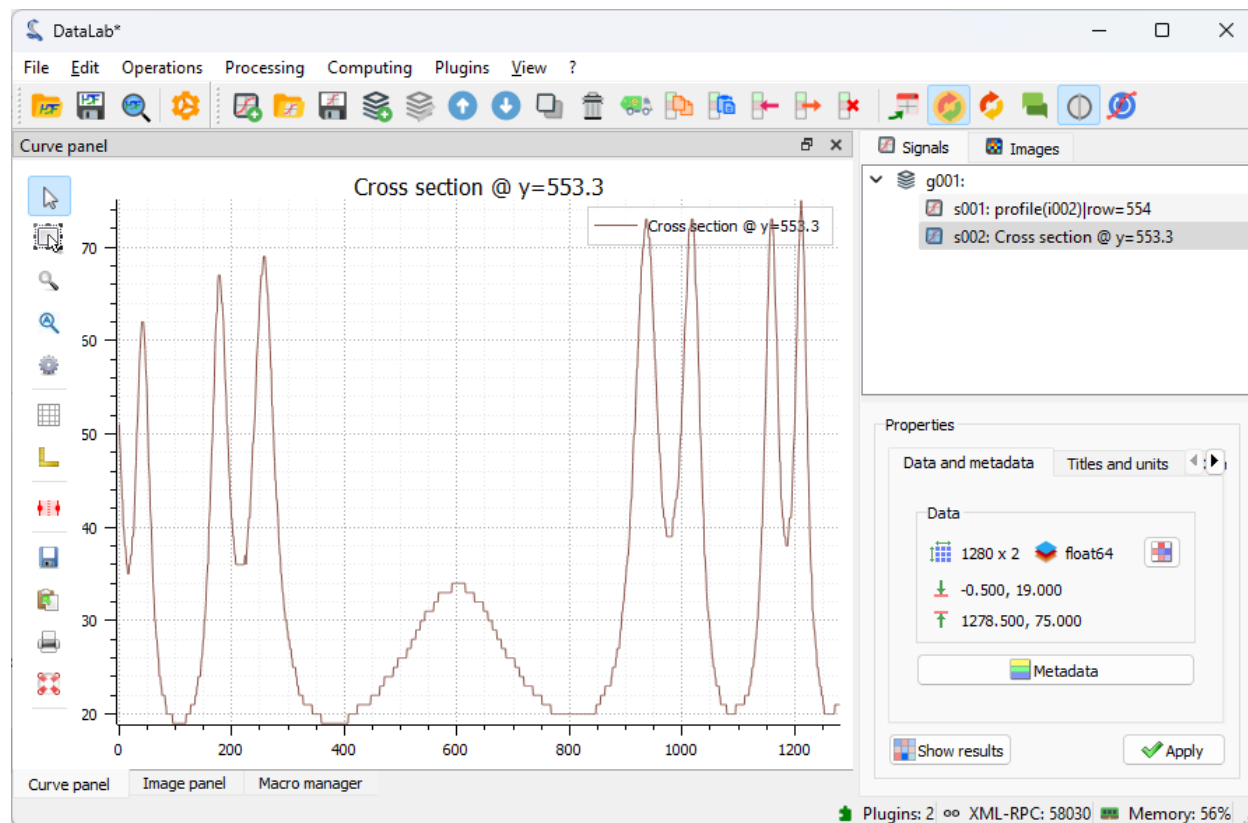


FIG. 67 – Le profil d'intensité est ajouté au panneau « Signaux », et DataLab bascule vers ce panneau pour afficher le profil.

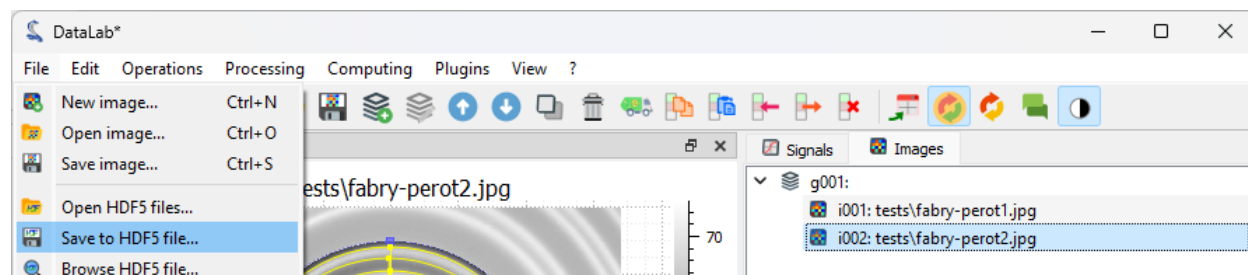


FIG. 68 – Sauvegardez l'espace de travail dans un fichier avec « Fichier > Sauvegarder dans un fichier HDF5... », ou le bouton dans la barre d'outils.

## Mesurer la taille d'un faisceau laser

Cet exemple montre comment mesurer la taille d'un faisceau laser le long de l'axe de propagation, en utilisant DataLab :

- Ouvrir toutes les images d'un dossier
- Appliquer un seuillage aux images
- Extraire le profil d'intensité le long d'une ligne horizontale
- Ajuster le profil d'intensité à une fonction gaussienne
- Calculer la largeur à mi-hauteur (FWHM) du profil d'intensité
- Essayer une autre méthode : extraire le profil d'intensité radial
- Calculer la FWHM du profil d'intensité radial
- Effectuer la même analyse sur une pile d'images et sur les profils résultants
- Tracer la taille du faisceau en fonction de la position le long de l'axe de propagation

Tout d'abord, nous ouvrons DataLab et chargeons les images :

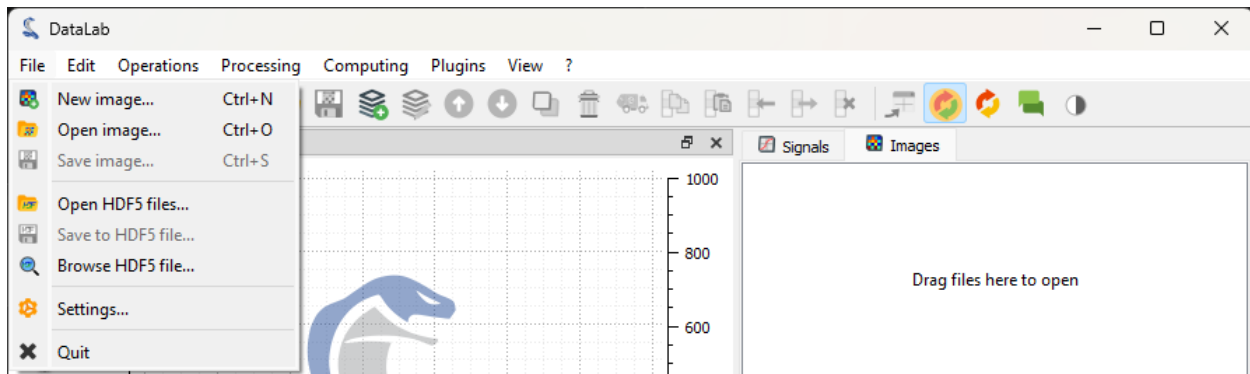


FIG. 69 – Ouvrir les fichiers d'images avec « Fichier > Ouvrir... », ou avec le bouton dans la barre d'outils, ou en faisant glisser et déposer les fichiers dans DataLab (sur le panneau de droite).

Les images sélectionnées sont chargées dans le panneau « Images ». La dernière image est affichée dans la fenêtre principale. Sur chaque image, nous pouvons zoomer en appuyant sur le bouton droit de la souris et en faisant glisser la souris vers le haut et vers le bas. Nous pouvons également déplacer l'image en appuyant sur le bouton du milieu de la souris et en faisant glisser la souris.

**Note :** Si nous voulons afficher les images côte à côte, nous pouvons sélectionner l'entrée « Distribuer sur une grille » dans le menu « Opérations ».

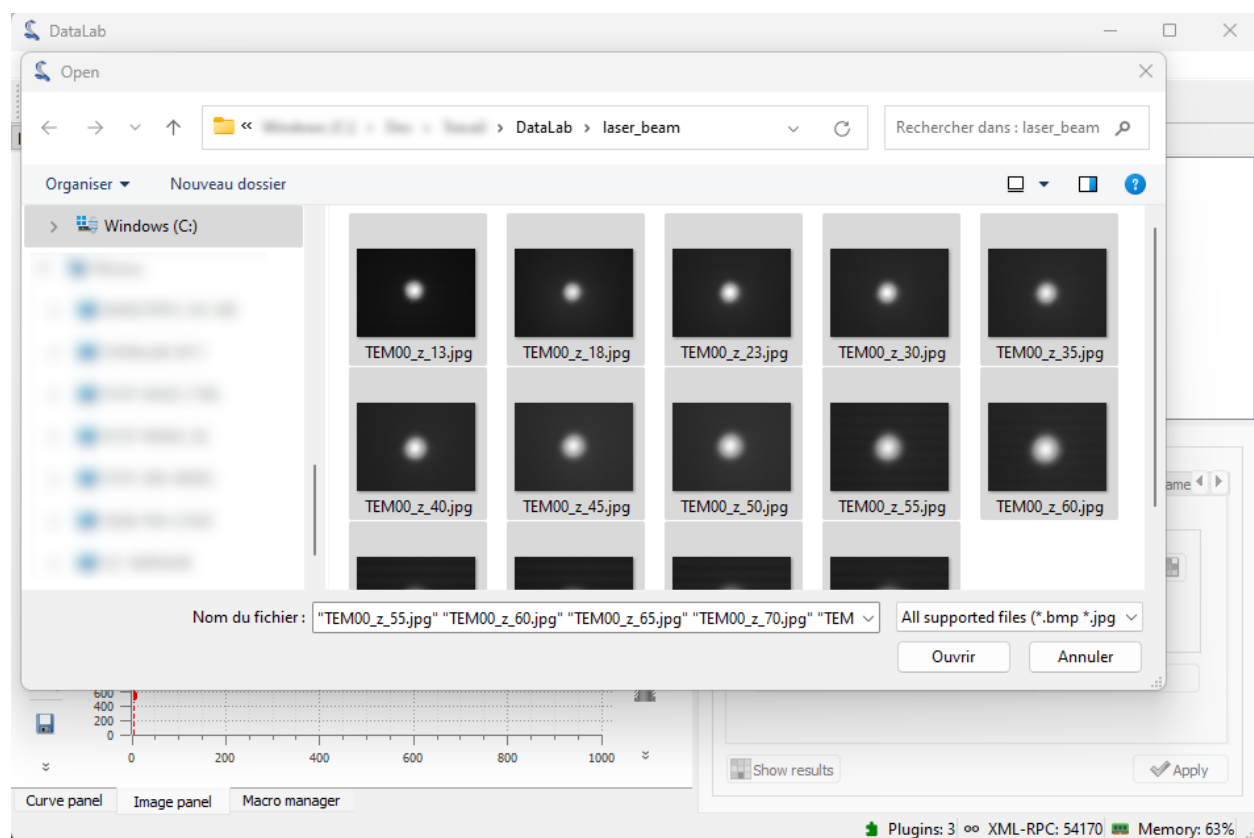


FIG. 70 – Sélectionnez les images de test « TEM00\_z\_\*.jpg » et cliquez sur « Ouvrir ».

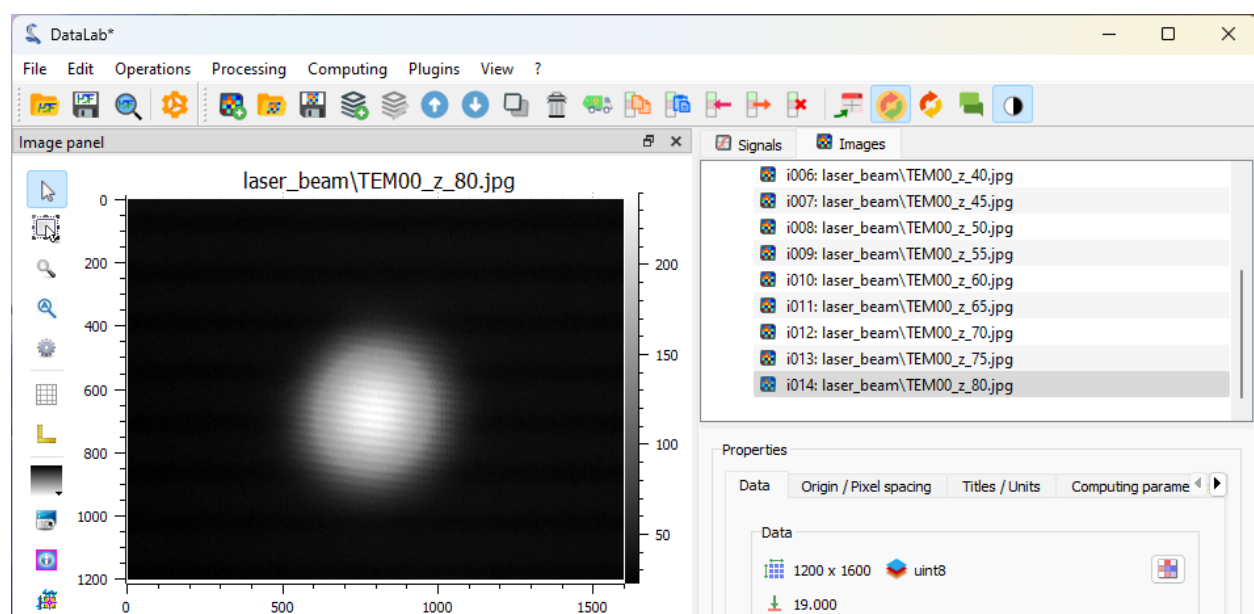


FIG. 71 – Zoomer avec le bouton droit de la souris. Déplacer l'image avec le bouton du milieu de la souris.

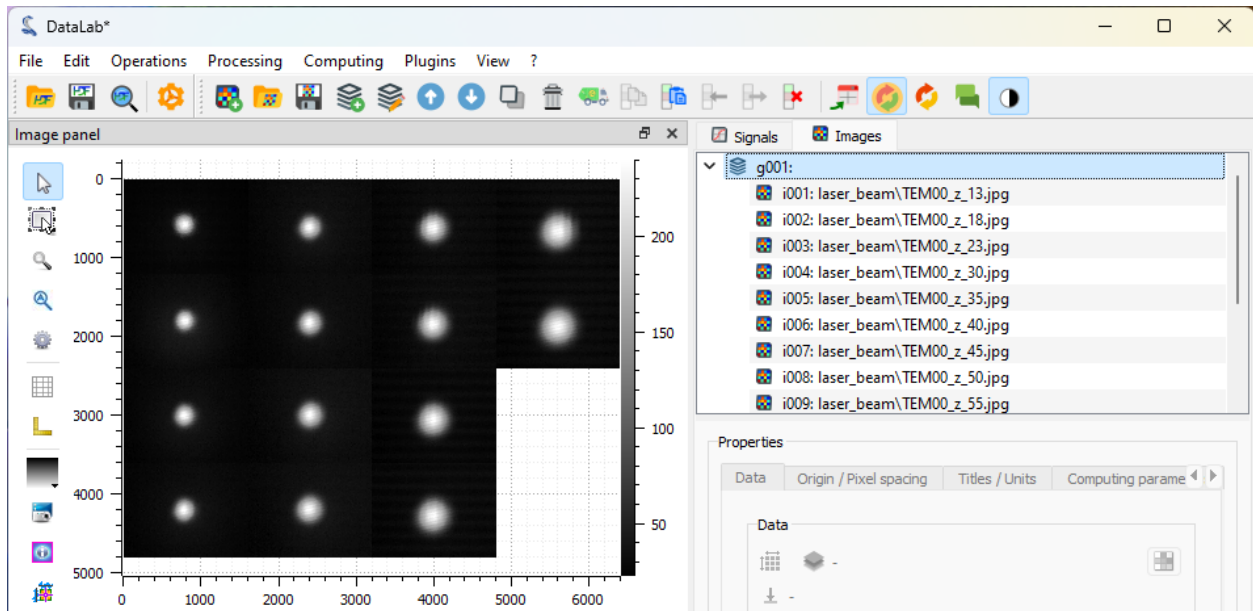


FIG. 72 – Images distribuées sur une grille de 4 lignes

Mais, revenons à l’affichage initial en sélectionnant l’entrée « Réinitialiser les positions des images » dans le menu « Opérations ».

Si nous sélectionnons l’une des images, nous pouvons voir qu’il y a du bruit de fond, il peut donc être utile d’appliquer un seuillage aux images.

Maintenant, essayons une autre méthode pour mesurer la taille du faisceau.

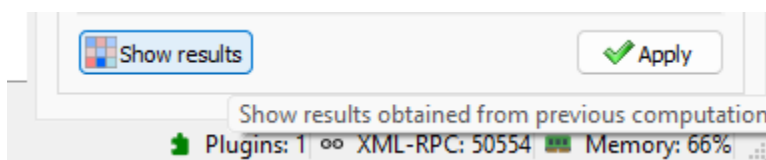
À partir du panneau « Images », nous pouvons extraire le profil d’intensité radial avec « Opérations > Profils d’intensité > Profil radial ».

Toutes ces opérations et calculs que nous avons effectués sur une seule image peuvent être appliqués à toutes les images du panneau « Images ».

Pour ce faire, nous commençons par nettoyer le panneau « Signaux » (avec « Édition > Tout supprimer » ou le bouton dans la barre d’outils). Nous nettoyons également les résultats intermédiaires dans le panneau « Images » en sélectionnant les images obtenues lors de notre prototypage et en les supprimant individuellement (avec « Édition > Supprimer » ou le bouton ).

Ensuite, nous sélectionnons toutes les images dans le panneau « Images » (individuellement, ou en sélectionnant l’ensemble du groupe « g001 »).

**Note :** Si vous souhaitez afficher à nouveau les résultats d’analyse, vous pouvez sélectionner l’entrée « Afficher les résultats » dans le menu Analyse », ou le bouton « Afficher les résultats », en dessous de la liste des images :



Enfin, nous pouvons enregistrer l’espace de travail dans un fichier . L’espace de travail contient toutes les images et signaux qui ont été chargés ou traités dans DataLab. Il contient également les résultats d’analyse, les paramètres de

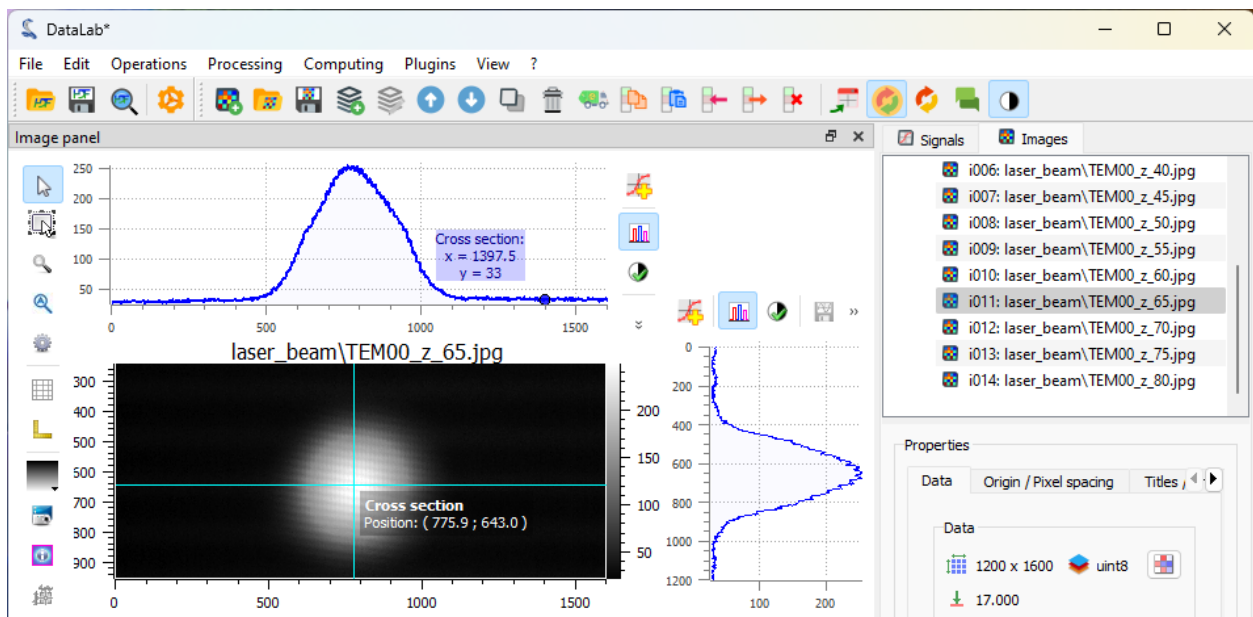


FIG. 73 – Sélectionnez l’une des images dans le panneau « Images », sélectionnez l’image associée dans le panneau de visualisation, et activez l’outil « Section transversale » dans la barre d’outils verticale à gauche du panneau de visualisation (cet outil est une fonctionnalité PlotPy). Sur cette figure, nous pouvons voir que le bruit de fond est d’environ 30 lsb (pour afficher le marqueur de courbe, nous avons dû sélectionner la courbe de profil et cliquer avec le bouton droit de la souris dessus pour afficher le menu contextuel, puis sélectionner « Marqueurs > Lié à l’élément actif »).

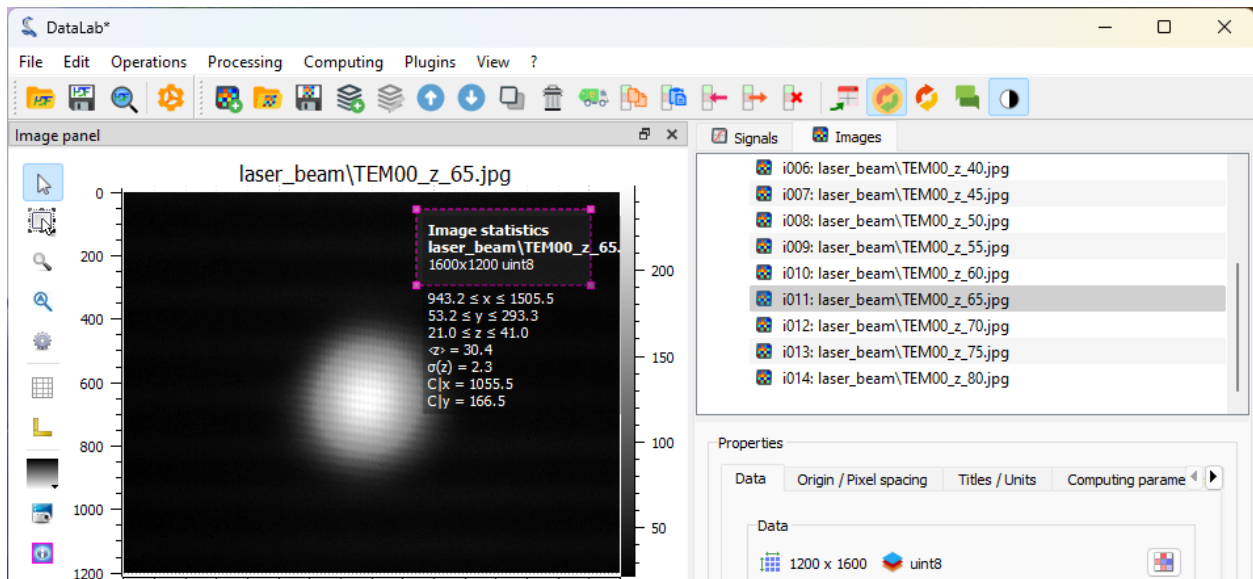


FIG. 74 – Une autre façon de mesurer le bruit de fond est d’utiliser l’outil « Statistiques de l’image » dans la barre d’outils verticale à gauche du panneau de visualisation. Il affiche des statistiques sur une zone rectangulaire définie en faisant glisser la souris sur l’image. Cela confirme que le bruit de fond est d’environ 30 lsb.

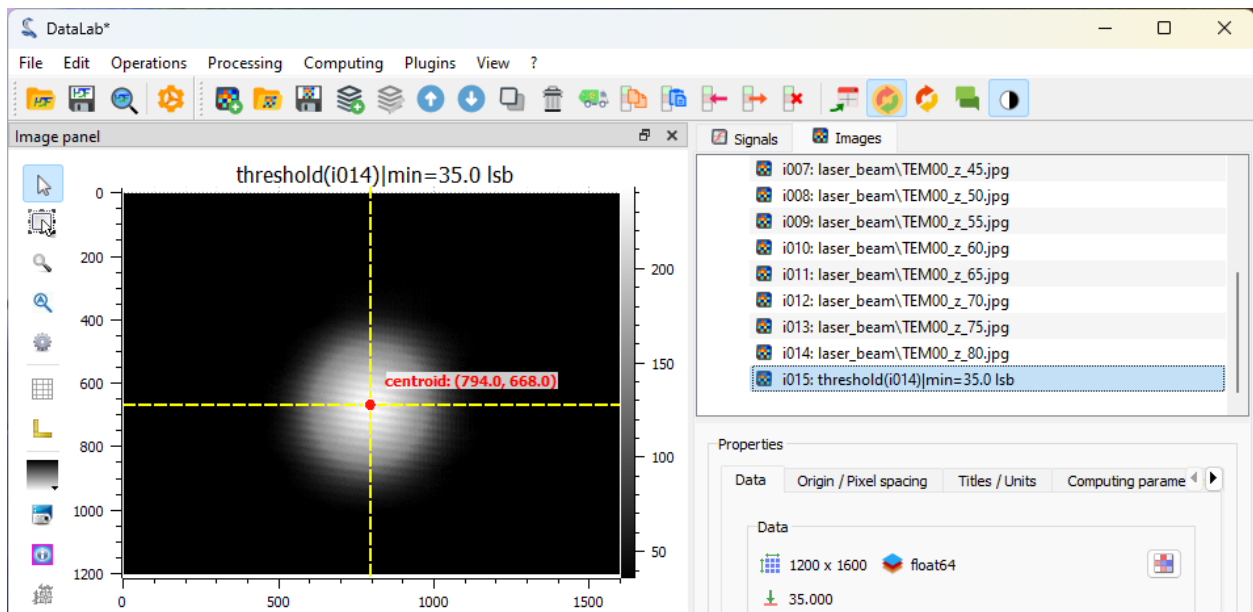


FIG. 75 – Après avoir appliqué un seuil à 35 lsb (avec « Traitement > Seuillage... »), nous pouvons calculer une position plus précise du centre du faisceau en utilisant Analyse > Centre de gravité ».

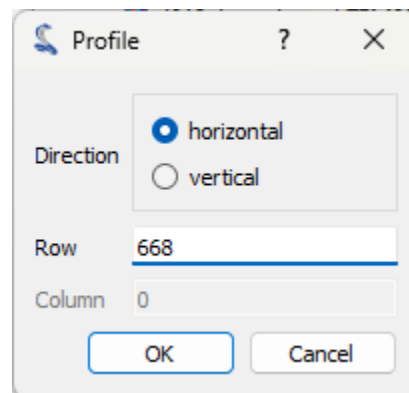


FIG. 76 – Ensuite, nous pouvons extraire un profil de ligne le long de l'axe horizontal avec « Opérations > Profils d'intensité > Profil de ligne ». Nous définissons la position de la ligne sur la position du centre de gravité calculée précédemment (c'est-à-dire 668).

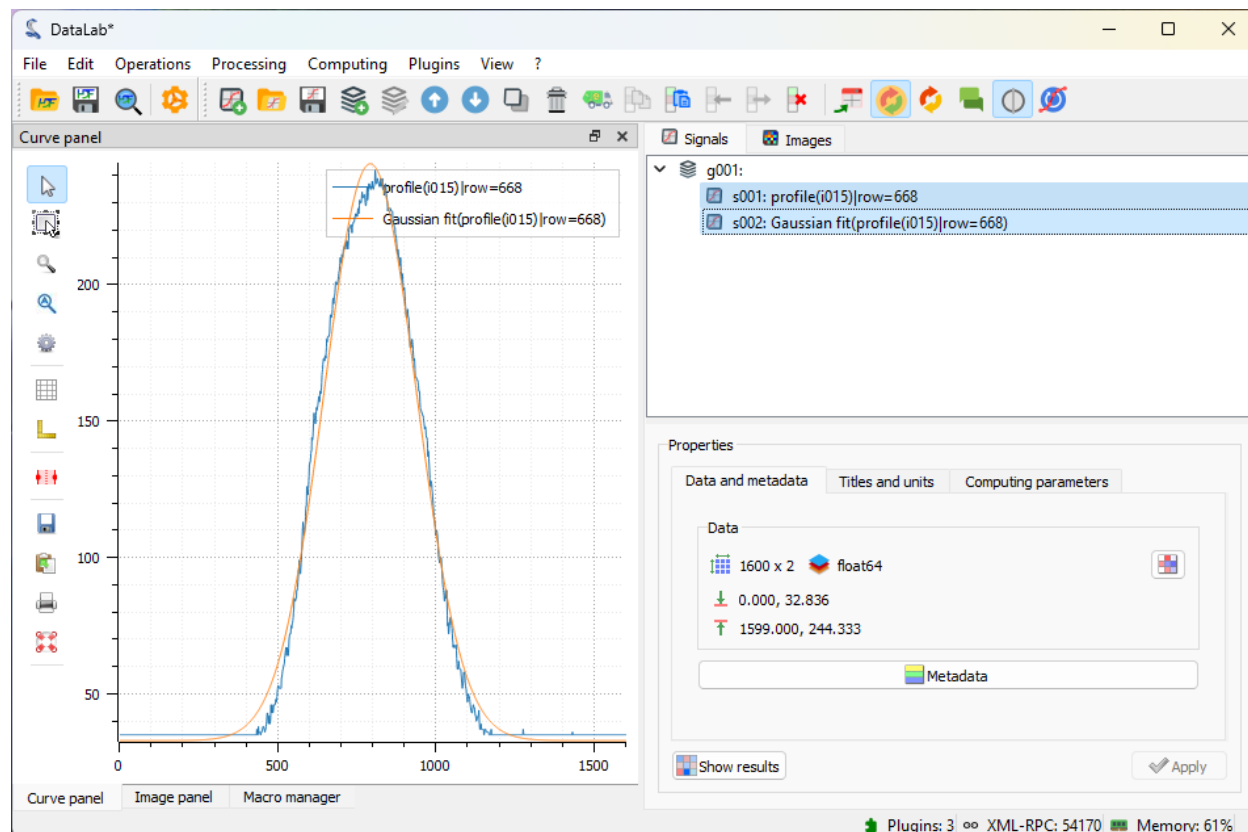


FIG. 77 – Le profil d'intensité est affiché dans le panneau « Signaux ». Nous pouvons ajuster le profil à une fonction gaussienne avec « Traitement > Ajustement > Ajustement gaussien ». Ici, nous avons sélectionné les deux signaux.

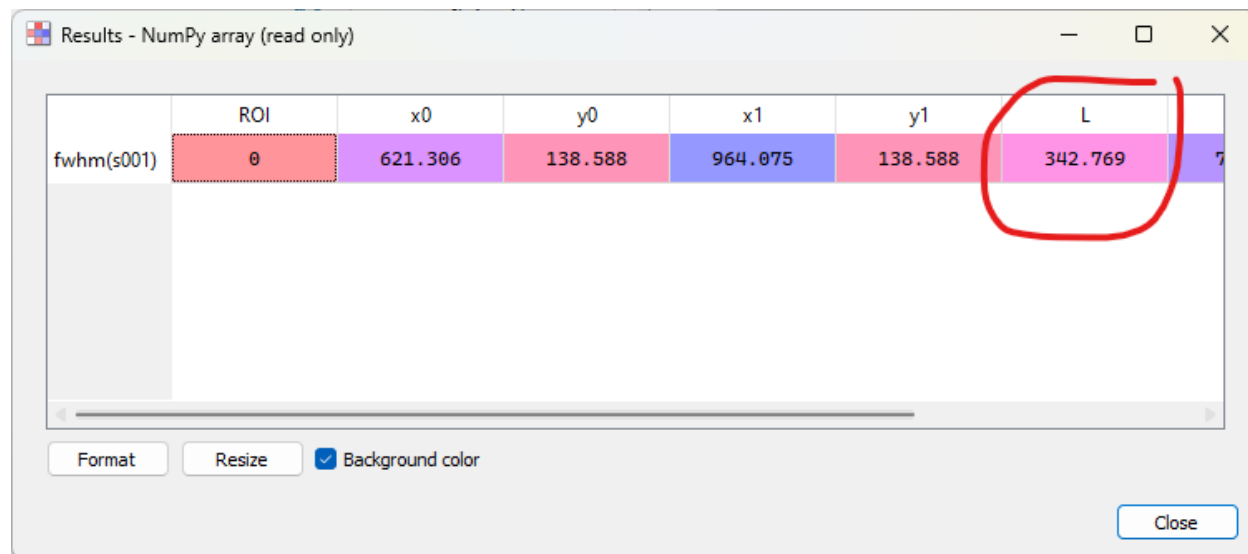


FIG. 78 – Si nous revenons au premier signal, le profil d'intensité, nous pouvons également calculer directement la FWHM avec Analyse > Largeur à mi-hauteur ». La boîte de dialogue « Résultats » affiche beaucoup d'informations sur le calcul, y compris la valeur FWHM (qui est la colonne L, « L » pour « Longueur » car la forme du résultat est un segment et FWHM est la longueur du segment).

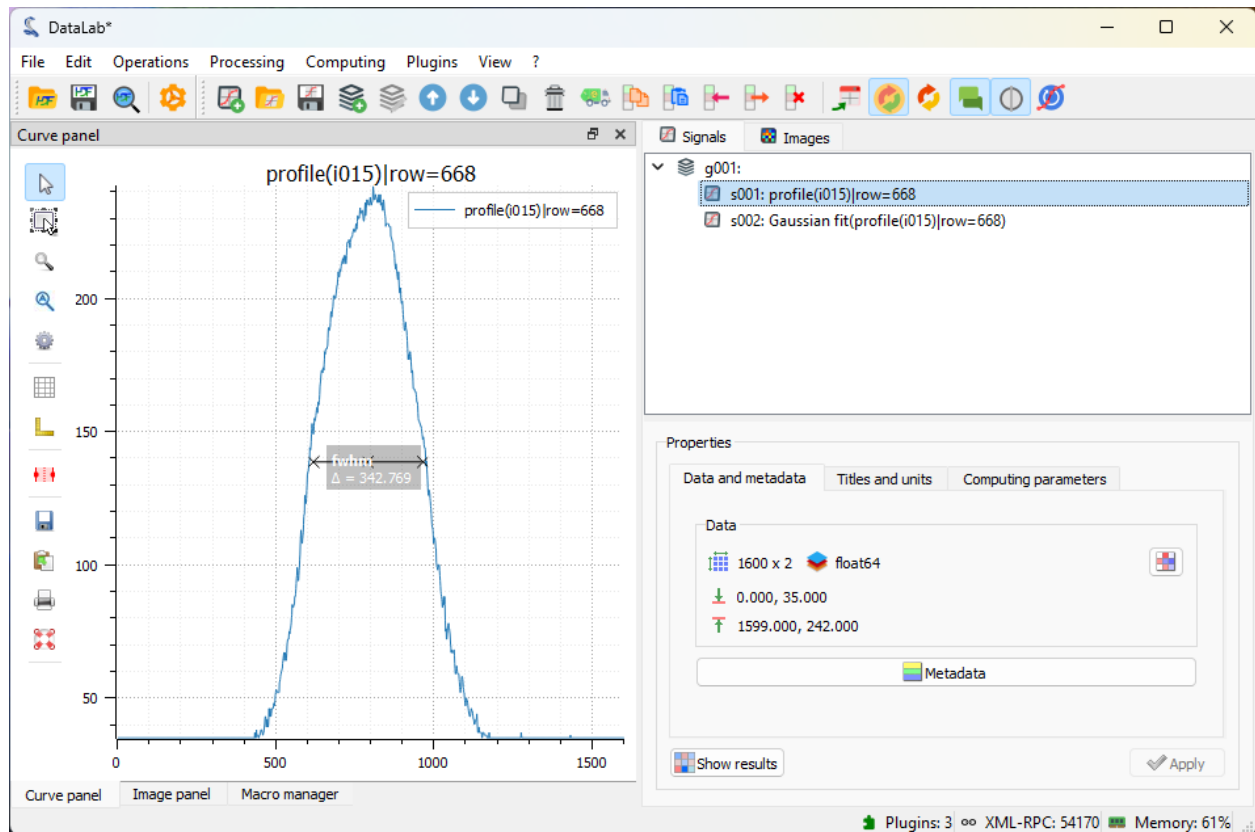


FIG. 79 – La largeur à mi-hauteur est également affichée sur la courbe, avec une étiquette optionnelle (ici, le titre de cette mesure a été affiché avec « Affichage > Afficher les titres des objets graphiques » ou le bouton dans la barre d'outils).

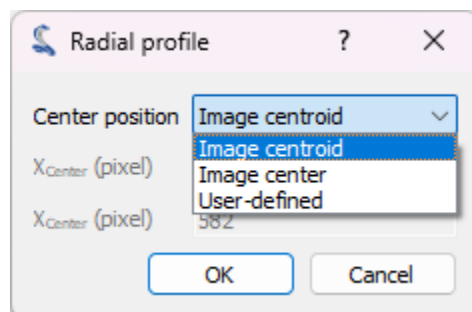


FIG. 80 – Le profil d'intensité radial peut être calculé autour de la position du centre de gravité, ou autour du centre de l'image, ou autour d'une position définie par l'utilisateur. Ici, nous avons sélectionné la position du centre de gravité.

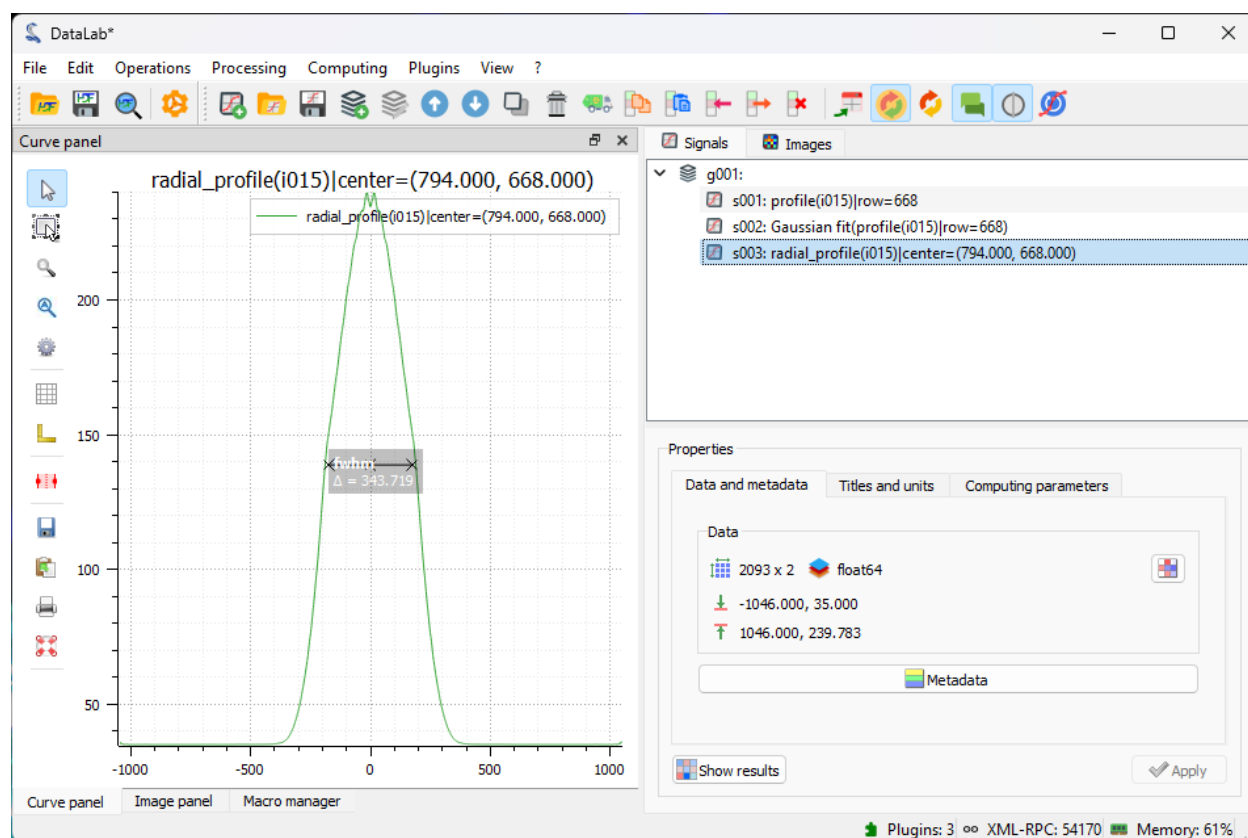


FIG. 81 – Le profil d'intensité radial est affiché dans le panneau « Signaux ». Il est plus lisse que le profil de ligne, car il est calculé à partir d'un plus grand nombre de pixels, ce qui permet de lisser le bruit.

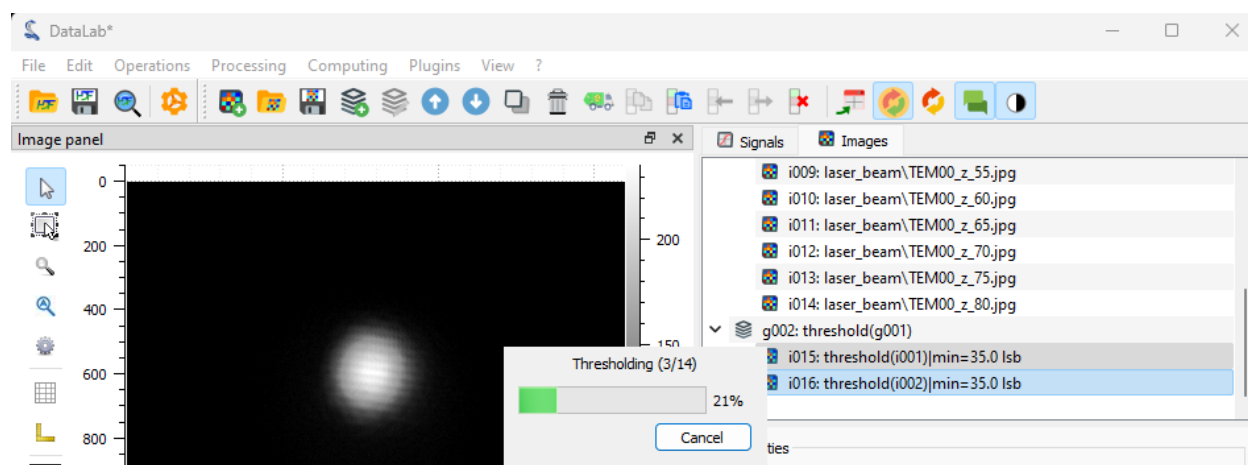


FIG. 82 – Nous appliquons le seuil à toutes les images, puis nous extrayons le profil d'intensité radial pour toutes les images (après avoir sélectionné l'ensemble du groupe « g002 » - il devrait être automatiquement sélectionné si vous aviez sélectionné « g001 » avant d'appliquer le seuil).

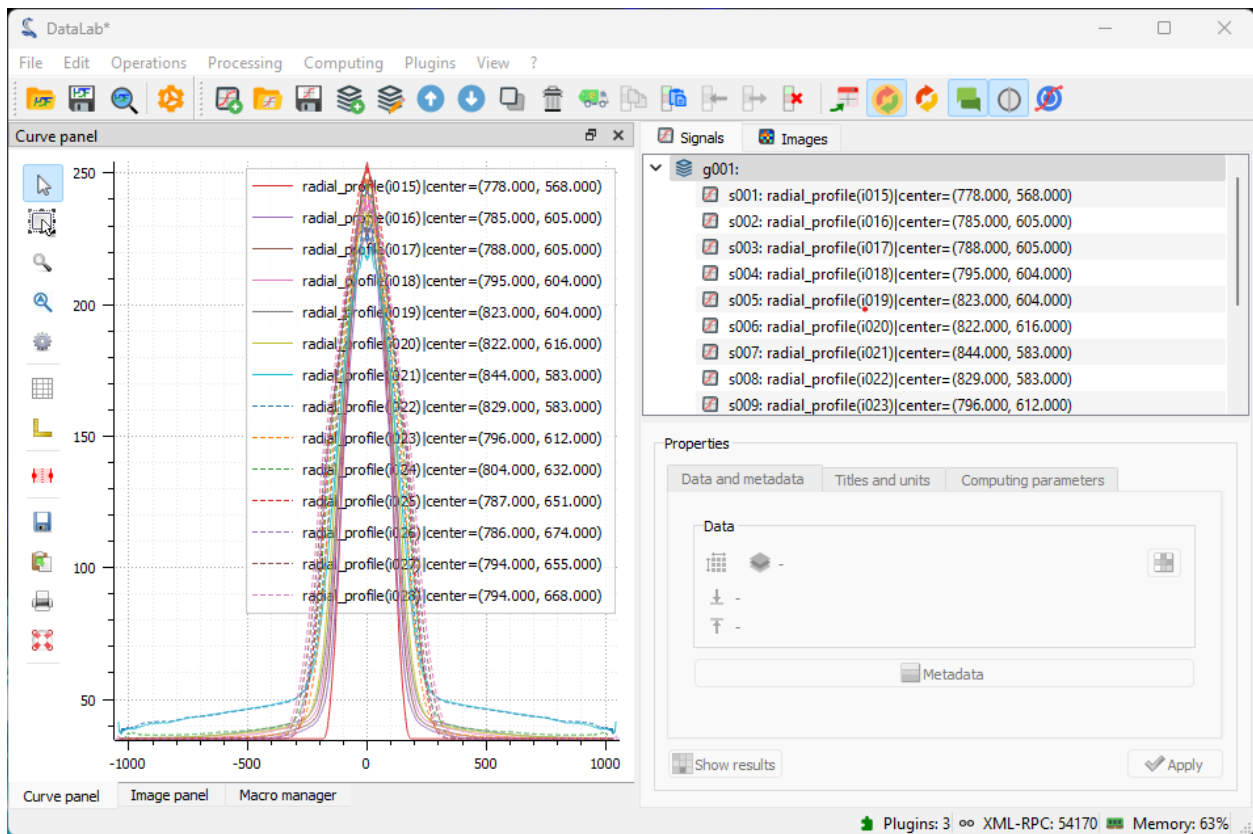


FIG. 83 – Le panneau « Signaux » contient maintenant tous les profils d'intensité radiaux.

Results - NumPy array (read only)

	ROI	x0	y0	x1	y1	L	Xc	Yc
fwhm(s001)	0	-90.146	146.086	90.1454	146.086	180.291	-0.000307247	146.086
fwhm(s002)	0	-97.5273	134.95	97.5273	134.95	195.055	-2.84217e-14	134.95
fwhm(s003)	0	-102.844	144.563	102.845	144.563	205.689	0.000226296	144.563
fwhm(s004)	0	-108.246	140.839	108.246	140.839	216.492	-1.66472e-05	140.839
fwhm(s005)	0	-114.898	134.226	114.898	134.226	229.796	-1.0366e-05	134.226
fwhm(s006)	0	-120.149	138.229	120.147	138.229	240.295	-0.000828078	138.229
fwhm(s007)	0	-132.471	134.219	132.471	134.219	264.942	2.84217e-14	134.219
fwhm(s008)	0	-136.738	138.216	136.737	138.216	273.475	-0.000167016	138.216
fwhm(s009)	0	-136.727	139.425	136.727	139.425	273.454	-0.000201918	139.425

Format    Resize    ☒ Background color

Close

FIG. 84 – Nous pouvons calculer la FWHM de tous les profils d'intensité radiaux : la boîte de dialogue « Résultats » affiche les valeurs FWHM pour tous les profils.

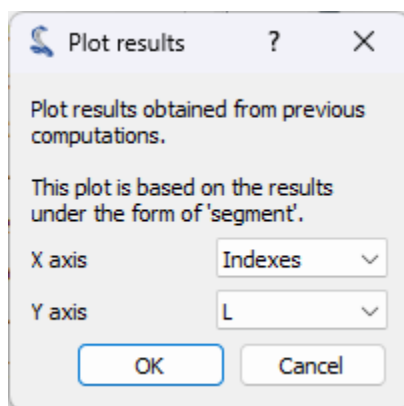


FIG. 85 – Enfin, nous pouvons tracer la taille du faisceau en fonction de la position le long de l’axe de propagation. Pour ce faire, nous utilisons la fonction « Tracer les résultats » dans le menu Analyse ». Cette fonction permet de tracer les ensembles de données de résultats en choisissant les axes x et y parmi les colonnes de résultats. Ici, nous choisissons de tracer les valeurs FWHM ( $L$ ) en fonction de l’index de l’image (*Indices*).

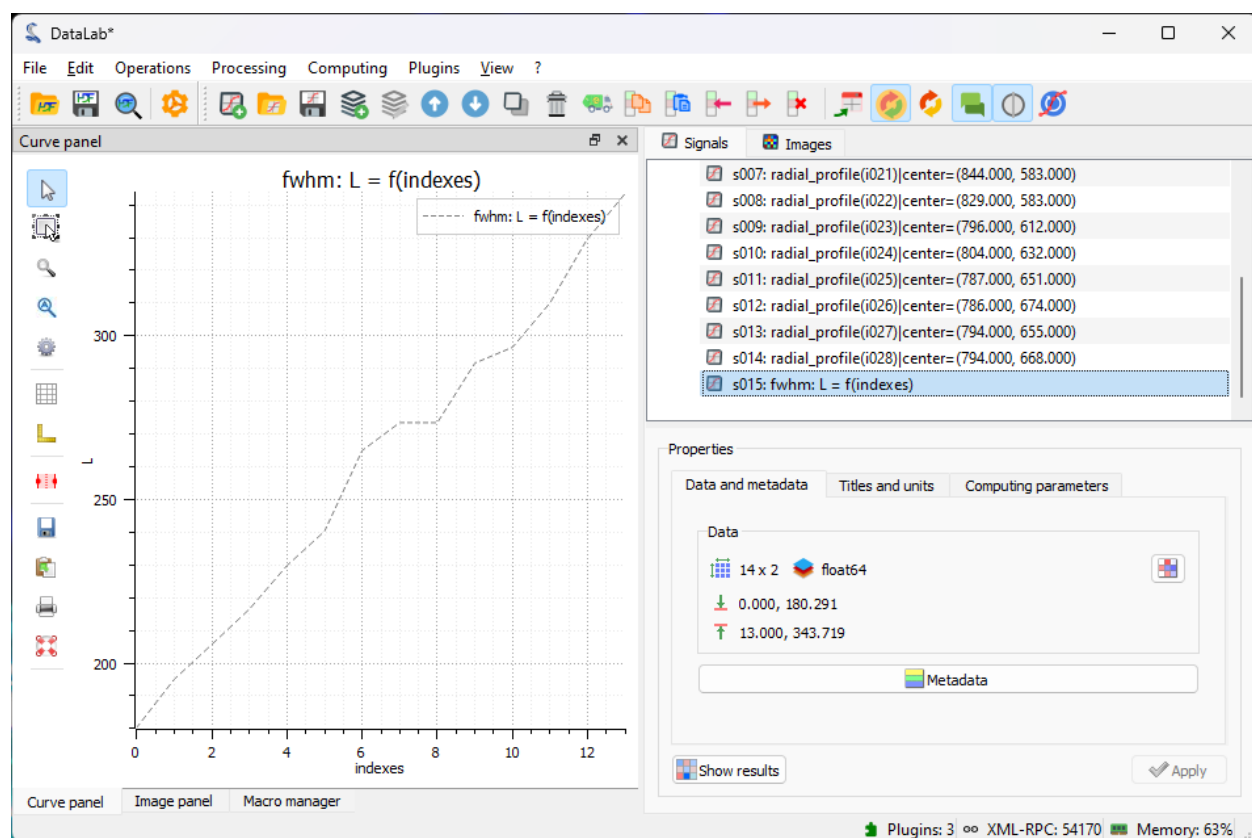


FIG. 86 – Le tracé est affiché dans le panneau « Signaux » et montre que la taille du faisceau augmente avec la position le long de l’axe de propagation (la position est ici en unités arbitraires, l’index de l’image).

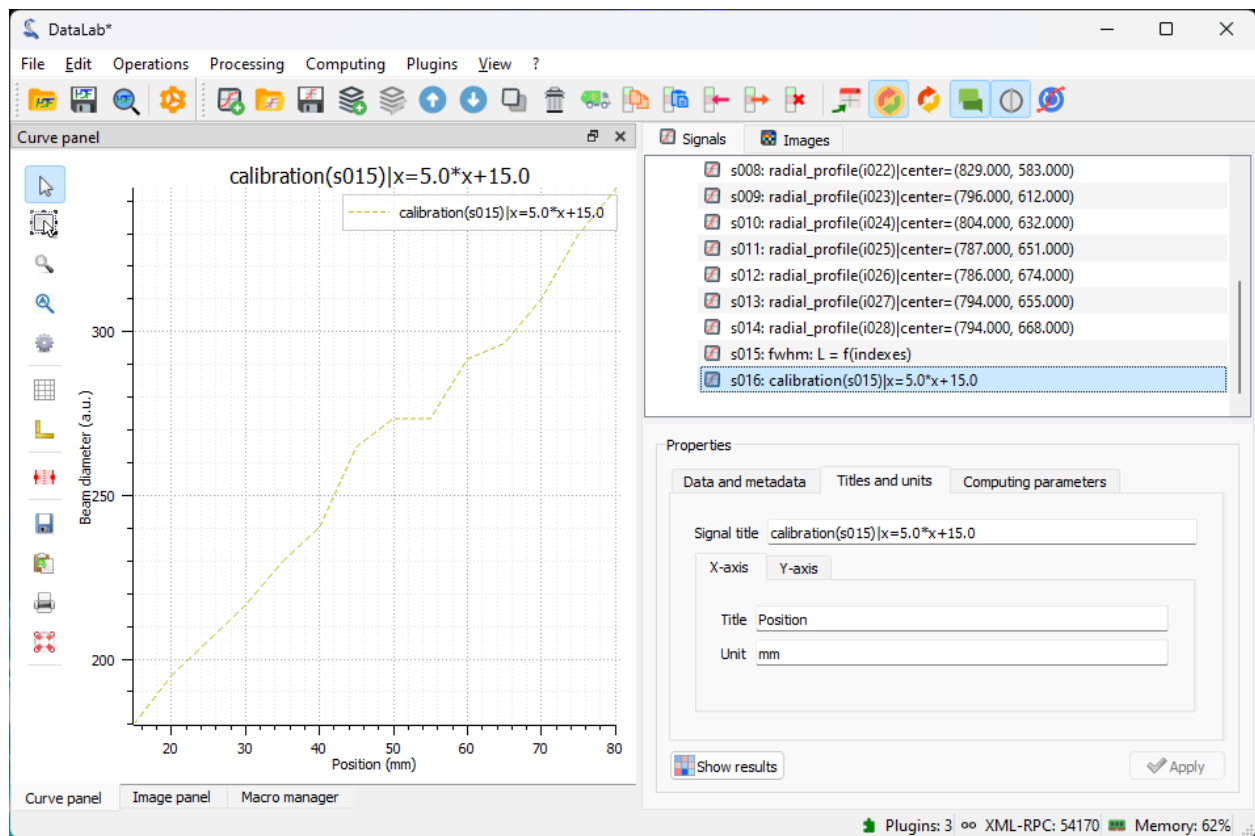




FIG. 87 – Nous pouvons également étalonner les axes X et Y en utilisant « Traitement > Étalonnage linéaire ». Ici, nous avons réglé l’axe X sur la position en mm (et entré le titre et l’unité dans le groupe de boîtes « Propriétés »).

visualisation (cartes de couleurs, contraste, etc.), les métadonnées et les annotations.

Si vous souhaitez charger à nouveau l'espace de travail, vous pouvez utiliser « Fichier > Ouvrir un fichier HDF5... » (ou le bouton  dans la barre d'outils) pour charger l'ensemble de l'espace de travail, ou « Fichier > Parcourir un fichier HDF5... » (ou le bouton  dans la barre d'outils) pour charger uniquement une sélection d'ensembles de données de l'espace de travail.

## Prototypage d'une chaîne de traitement personnalisée

Cet exemple montre comment prototyper une chaîne de traitement d'image personnalisée en utilisant DataLab :

- Définir une fonction de traitement personnalisée
- Créer une macro-commande pour appliquer la fonction à une image
- Utiliser le même code à partir d'un IDE externe (par exemple Spyder) ou d'un notebook Jupyter
- Créer un plugin pour intégrer la fonction dans l'interface graphique de DataLab

## Définir une fonction de traitement personnalisée

Pour illustrer l'extensibilité de DataLab, nous utiliserons une fonction de traitement d'image simple qui n'est pas disponible dans la distribution standard de DataLab, et qui représente un cas d'utilisation typique pour le prototypage d'une chaîne de traitement personnalisée.

La fonction sur laquelle nous allons travailler est un filtre de débruitage qui combine les idées de moyennage et de détection de contours. Ce filtre va moyenner les valeurs de pixels dans le voisinage, mais avec une particularité : il donnera moins de poids aux pixels qui sont significativement différents du pixel central, en supposant qu'ils pourraient faire partie d'un bord ou d'un bruit.

Voici le code de la fonction `weighted_average_denoise` :

```
def weighted_average_denoise(data: np.ndarray) -> np.ndarray:
    """Apply a custom denoising filter to an image.

    This filter averages the pixels in a 5x5 neighborhood, but gives less weight
    to pixels that significantly differ from the central pixel.
    """

    def filter_func(values: np.ndarray) -> float:
        """Filter function"""
        central_pixel = values[len(values) // 2]
        differences = np.abs(values - central_pixel)
        weights = np.exp(-differences / np.mean(differences))
        return np.average(values, weights=weights)

    return spi.generic_filter(data, filter_func, size=5)
```

Pour tester notre fonction de traitement, nous utiliserons une image générée à partir d'un exemple de plugin DataLab (plugins/examples/cdl\_example\_imageproc.py). Avant de commencer, assurez-vous que le plugin est installé dans DataLab (voir les premières étapes du tutoriel [Détection de taches sur une image](#)).

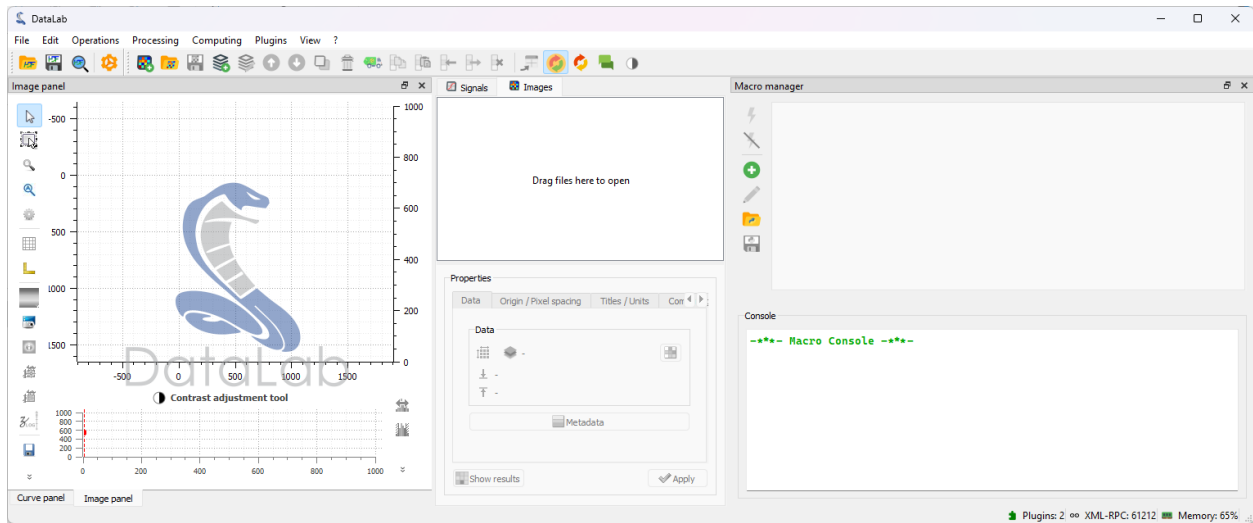


FIG. 88 – Pour commencer, nous réorganisons la disposition de la fenêtre de DataLab pour avoir le « Panneau Image » à gauche et le « Gestionnaire de Macros » à droite.

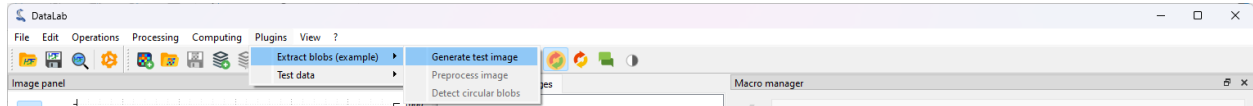


FIG. 89 – Nous générons une nouvelle image en utilisant le menu « Plugins > Extract blobs (example) > Generate test image ».

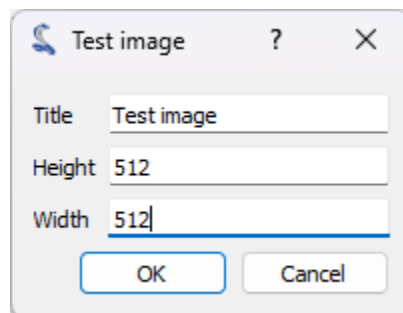


FIG. 90 – Nous sélectionnons une taille limitée pour l'image (par exemple 512x512 pixels) car notre algorithme est assez lent, et cliquons sur « OK ».

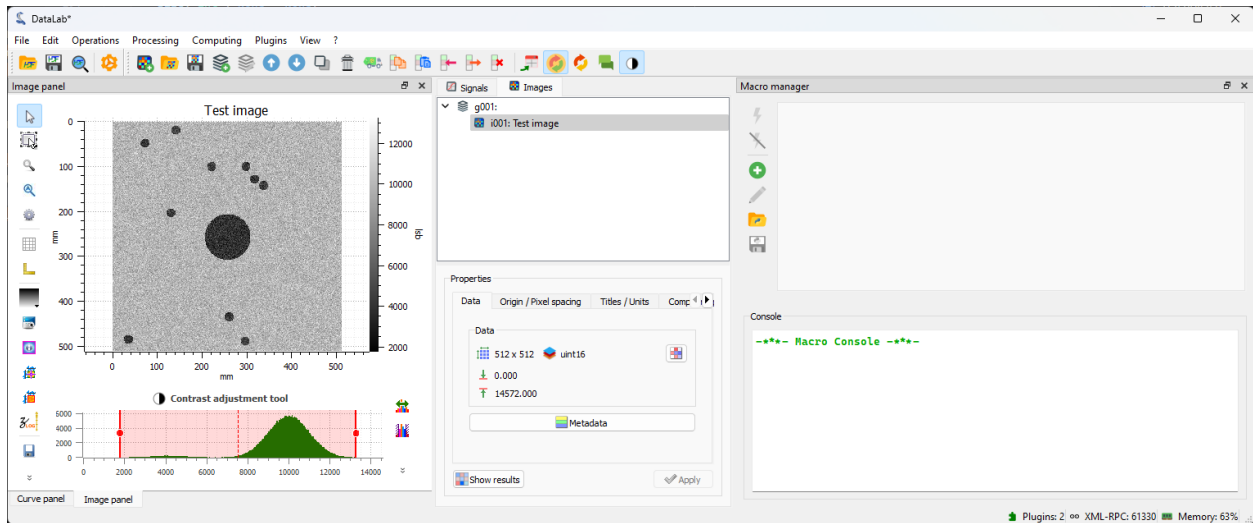


FIG. 91 – Nous pouvons maintenant voir l'image générée dans le « Panneau Image ».

## Créer une macro-commande

Revenons à notre fonction personnalisée. Nous pouvons créer une nouvelle macro-commande qui appliquera la fonction à l'image actuelle. Pour ce faire, nous ouvrons le « Gestionnaire de Macros » et cliquons sur le bouton « Nouvelle macro ».

DataLab crée une nouvelle macro-commande qui n'est pas vide : elle contient un code d'exemple qui montre comment créer une nouvelle image et l'ajouter au « Panneau Image ». Nous pouvons supprimer ce code et le remplacer par le nôtre :

```
# Import the necessary modules
import numpy as np
import scipy.ndimage as spi
from cdl.proxy import RemoteProxy

# Define our custom processing function
def weighted_average_denoise(values: np.ndarray) -> float:
    """Apply a custom denoising filter to an image.

    This filter averages the pixels in a 5x5 neighborhood, but gives less weight
    to pixels that significantly differ from the central pixel.
    """
    central_pixel = values[len(values) // 2]
    differences = np.abs(values - central_pixel)
    weights = np.exp(-differences / np.mean(differences))
    return np.average(values, weights=weights)

# Initialize the proxy to DataLab
proxy = RemoteProxy()

# Switch to the "Image Panel" and get the current image
proxy.set_current_panel("image")
image = proxy.get_object()
if image is None:
```

(suite sur la page suivante)

(suite de la page précédente)

```
# We raise an explicit error if there is no image to process
raise RuntimeError("No image to process!")

# Get a copy of the image data, and apply the function to it
data = np.array(image.data, copy=True)
data = spi.generic_filter(data, weighted_average_denoise, size=5)

# Add new image to the panel
proxy.add_image("My custom filtered data", data)
```

Dans DataLab, les macro-commandes sont simplement des scripts Python :

- Les macros font partie de l'**espace de travail** de DataLab, ce qui signifie qu'elles sont sauvegardées et restaurées lors de l'exportation et de l'importation vers/depuis un fichier HDF5.
- Les macros sont exécutées dans un processus séparé, nous devons donc importer les modules nécessaires et initialiser le proxy vers DataLab. Le proxy est un objet spécial qui permet de communiquer avec DataLab.
- En conséquence, **lors de la définition d'un plugin ou du contrôle de DataLab à partir d'un IDE externe, nous pouvons utiliser exactement le même code que dans la macro-commande.** C'est un point très important, car cela signifie que nous pouvons prototyper notre chaîne de traitement dans DataLab, puis utiliser le même code dans un plugin ou dans un IDE externe pour le développer davantage.

**Note :** La macro-commande est exécutée dans l'environnement Python de DataLab, nous pouvons donc utiliser les modules disponibles dans DataLab. Cependant, nous pouvons également utiliser nos propres modules, tant qu'ils sont installés dans l'environnement Python de DataLab ou dans une distribution Python compatible avec l'environnement Python de DataLab.

Si vos modules personnalisés ne sont pas installés dans l'environnement Python de DataLab, et s'ils sont compatibles avec la version Python de DataLab, vous pouvez préfixer `sys.path` avec le chemin vers la distribution Python qui contient vos modules :

```
import sys
sys.path.insert(0, "/path/to/my/python/distribution")
```

Cela vous permettra d'importer vos modules dans la macro-commande et de les mélanger avec les modules disponibles dans DataLab.

**Avertissement :** Si vous utilisez cette méthode, assurez-vous que vos modules sont compatibles avec la version Python de DataLab. Sinon, vous obtiendrez des erreurs lors de leur importation.

Maintenant, exécutons la macro-commande en cliquant sur le bouton « Exécuter la macro » :

- La macro-commande est exécutée dans un processus séparé, nous pouvons donc continuer à travailler dans DataLab pendant que la macro-commande s'exécute. Et, si la macro-commande prend trop de temps à s'exécuter, nous pouvons l'arrêter en cliquant sur le bouton « Arrêter la macro ».
- Pendant l'exécution de la macro-commande, nous pouvons voir la progression dans la fenêtre « Gestionnaire de Macros » : la sortie standard du processus est affichée dans la « Console » en dessous de l'éditeur de macro. Nous pouvons voir les messages suivants :
  - `---[...]---[# ==> Running 'Untitled 01' macro...]` : la macro-commande démarre
  - `Connecting to DataLab XML-RPC server...OK [...]` : le proxy est connecté à DataLab
  - `---[...]---[# <== 'Untitled 01' macro has finished]` : la macro-commande se termine

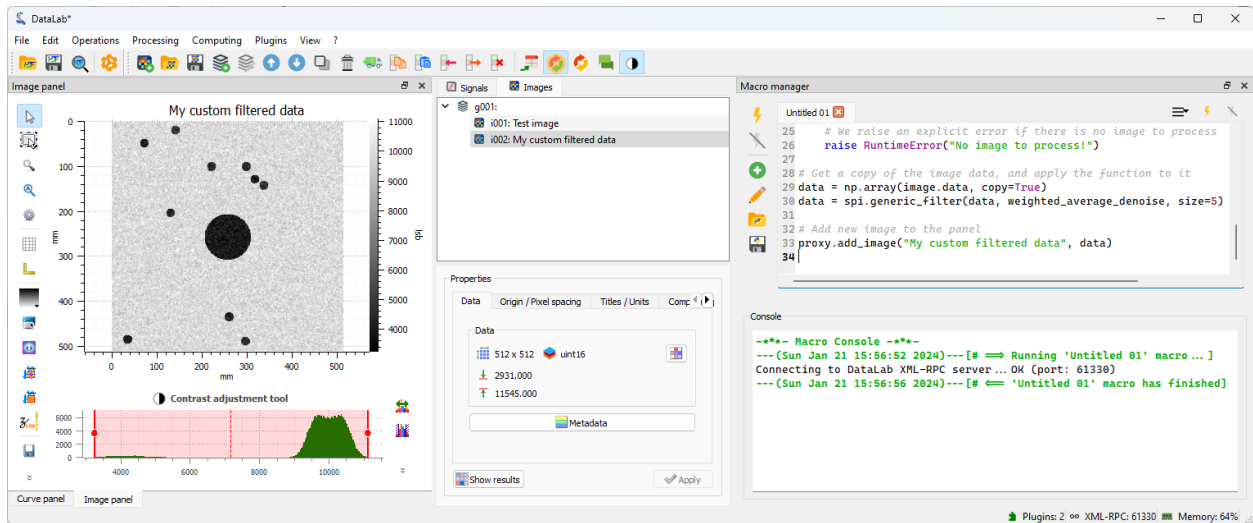


FIG. 92 – Lorsque la macro-commande est terminée, nous pouvons voir la nouvelle image dans le « Panneau Image ». Notre filtre a été appliqué à l'image, et nous pouvons voir que le bruit a été réduit.

## Prototypage avec un IDE externe

Maintenant que nous avons un prototype fonctionnel de notre chaîne de traitement, nous pouvons utiliser le même code dans un IDE externe pour le développer davantage.

Par exemple, nous pouvons utiliser l'IDE Spyder pour déboguer notre code. Pour ce faire, nous devons installer Spyder mais pas nécessairement dans l'environnement Python de DataLab (dans le cas de la version autonome de DataLab, ce ne serait de toute façon pas possible).

Le seul prérequis est d'installer un client DataLab dans l'environnement Python de Spyder :

- Si vous utilisez la version autonome de DataLab ou si vous voulez ou devez garder DataLab et Spyder dans des environnements Python séparés, vous pouvez installer le [DataLab Simple Client](#) (`cdl-client`) en utilisant le gestionnaire de paquets `pip` :

```
pip install cdl-client
```

Ou vous pouvez également installer le [paquet Python DataLab](#) (`cdl`) qui inclut le client (mais aussi d'autres modules, donc nous ne recommandons pas cette méthode si vous n'avez pas besoin de toutes les fonctionnalités de DataLab dans cet environnement Python) :

```
pip install cdl
```

- Si vous utilisez le paquet Python DataLab, vous pouvez exécuter Spyder dans le même environnement Python que DataLab, vous n'avez donc pas besoin d'installer le client : il est déjà disponible dans le paquet principal DataLab (le paquet `cdl`).

Une fois le client installé, nous pouvons démarrer Spyder et créer un nouveau script Python :

```
1 # -*- coding: utf-8 -*-
2 """
3 Example of remote control of DataLab current session,
4 from a Python script running outside DataLab (e.g. in Spyder)
5
6 Created on Fri May 12 12:28:56 2023
7
```

(suite sur la page suivante)

(suite de la page précédente)

```

8  @author: p.raybaut
9  """
10
11 # %% Importing necessary modules
12
13 import numpy as np
14 import scipy.ndimage as spi
15 from cdlclient import SimpleRemoteProxy
16
17 # %% Connecting to DataLab current session
18
19 proxy = SimpleRemoteProxy()
20 proxy.connect()
21
22 # %% Executing commands in DataLab (...)
23
24
25 # Define our custom processing function
26 def weighted_average_denoise(data: np.ndarray) -> np.ndarray:
27     """Apply a custom denoising filter to an image.
28
29     This filter averages the pixels in a 5x5 neighborhood, but gives less weight
30     to pixels that significantly differ from the central pixel.
31     """
32
33     def filter_func(values: np.ndarray) -> float:
34         """Filter function"""
35         central_pixel = values[len(values) // 2]
36         differences = np.abs(values - central_pixel)
37         weights = np.exp(-differences / np.mean(differences))
38         return np.average(values, weights=weights)
39
40     return spi.generic_filter(data, filter_func, size=5)
41
42
43 # Switch to the "Image Panel" and get the current image
44 proxy.set_current_panel("image")
45 image = proxy.get_object()
46 if image is None:
47     # We raise an explicit error if there is no image to process
48     raise RuntimeError("No image to process!")
49
50 # Get a copy of the image data, and apply the function to it
51 data = np.array(image.data, copy=True)
52 data = weighted_average_denoise(data)
53
54 # Add new image to the panel
55 proxy.add_image("Filtered using Spyder", data)

```

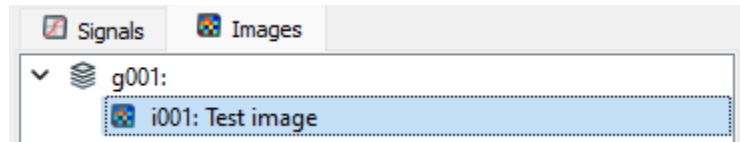


FIG. 93 – Nous retournons à DataLab et sélectionnons la première image dans le « Panneau Image ».

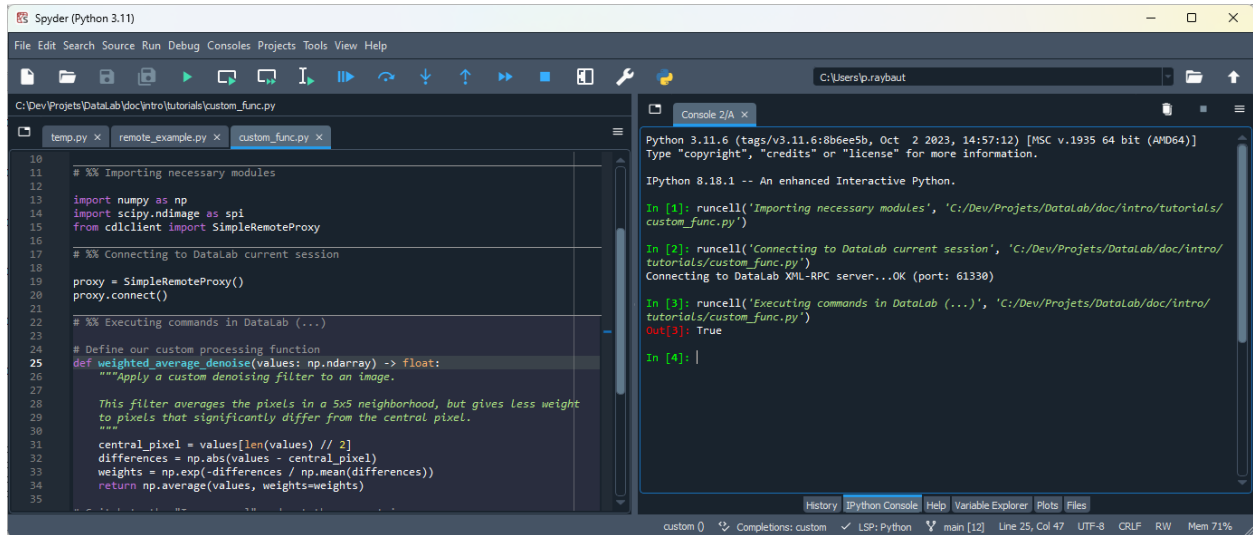


FIG. 94 – Ensuite, nous exécutons le script dans Spyder, étape par étape (en utilisant les cellules définies), et nous pouvons voir le résultat dans DataLab.

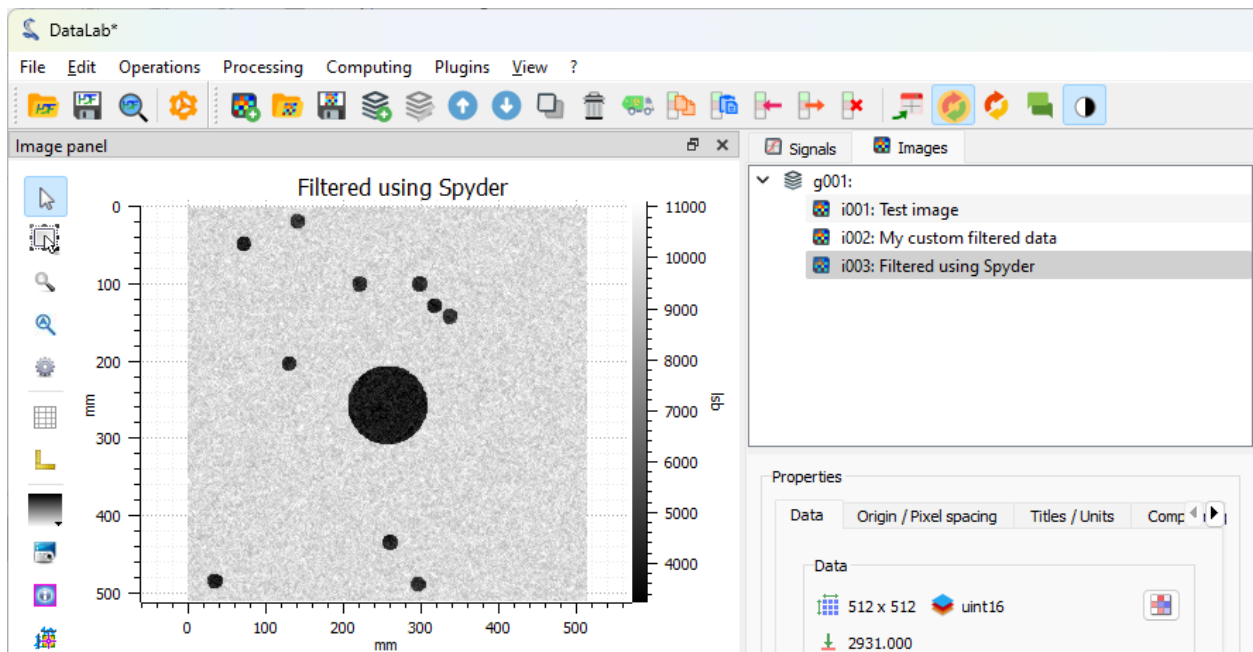


FIG. 95 – Nous pouvons voir dans DataLab qu'une nouvelle image a été ajoutée au « Panneau Image ». Cette image est le résultat de l'exécution du script dans Spyder. Ici, nous avons utilisé le script sans aucune modification, mais nous aurions pu le modifier pour tester de nouvelles idées, puis utiliser le script modifié dans DataLab.

## Prototypage avec un notebook Jupyter

Nous pouvons également utiliser un notebook Jupyter pour prototyper notre chaîne de traitement. Pour ce faire, nous devons installer Jupyter mais pas nécessairement dans l'environnement Python de DataLab (dans le cas de la version autonome de DataLab, ce ne serait de toute façon pas possible).

Le seul prérequis est d'installer un client DataLab dans l'environnement Python de Jupyter (voir la section précédente pour plus de détails : c'est exactement la même procédure que pour Spyder ou tout autre IDE comme Visual Studio Code, par exemple).

### DataLab custom function example

This is part of the DataLab's custom function tutorial which aims at illustrating the extensibility of DataLab (macros, plugins, and control from an external IDE or a Jupyter notebook).

The only requirement is to install the *DataLab Simple Client* package (using `pip install cdclient`, for example).

```
[2]: import numpy as np
import scipy.ndimage as spi
from cdclient import SimpleRemoteProxy

# Define our custom processing function
def weighted_average_denoise(values: np.ndarray) -> float:
    """Apply a custom denoising filter to an image.

    This filter averages the pixels in a 5x5 neighborhood, but gives less weight
    to pixels that significantly differ from the central pixel.
    """
    central_pixel = values[len(values) // 2]
    differences = np.abs(values - central_pixel)
    weights = np.exp(-differences / np.mean(differences))
    return np.average(values, weights=weights)
```

Connecting to DataLab current session:

```
[3]: proxy = SimpleRemoteProxy()
proxy.connect()

Connecting to DataLab XML-RPC server...OK (port: 61330)

Switch to the "Image panel" and get the current image
```

```
[5]: proxy.set_current_panel("image")
image = proxy.get_object()
if image is None:
    # We raise an explicit error if there is no image to process
    raise RuntimeError("No image to process!")
```

Get a copy of the image data, apply the function to it, and add new image to the panel

```
[6]: data = np.array(image.data, copy=True)
data = spi.generic_filter(data, weighted_average_denoise, size=5)
proxy.add_image("Filtered using a Jupyter notebook", data)
```

FIG. 96 – Une fois le client installé, nous pouvons démarrer Jupyter et créer un nouveau notebook.

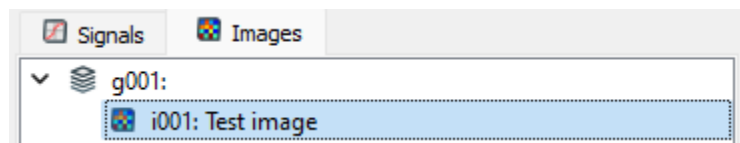


FIG. 97 – Nous retournons à DataLab et sélectionnons la première image dans le « Panneau Image ».

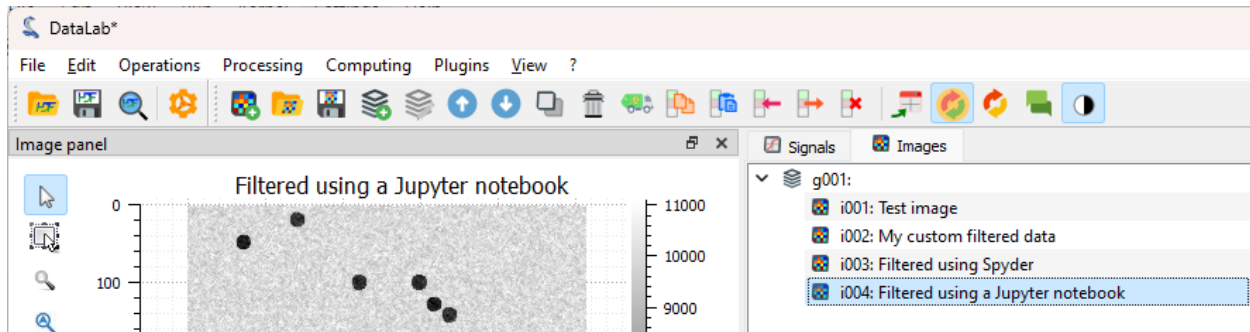


Fig. 98 – Ensuite, nous exécutons le notebook dans Jupyter, étape par étape (en utilisant les cellules définies), et nous pouvons voir le résultat dans DataLab. Une fois de plus, nous pouvons voir dans DataLab qu’une nouvelle image a été ajoutée au « Panneau Image ». Cette image est le résultat de l’exécution du notebook dans Jupyter. Comme pour le script dans Spyder, nous aurions pu modifier le notebook pour tester de nouvelles idées, puis utiliser le notebook modifié dans DataLab.

## Création d’un plugin

Maintenant que nous avons un prototype fonctionnel de notre chaîne de traitement, nous pouvons créer un plugin pour l’intégrer dans l’interface graphique de DataLab. Pour ce faire, nous devons créer un nouveau module Python qui contiendra le code du plugin. Nous pouvons utiliser le même code que dans la macro-commande, mais nous devons apporter quelques modifications.

### Voir aussi :

Le système de plugins est décrit dans la section [Plugins](#).

Mis à part l’intégration de la fonctionnalité à l’interface graphique de DataLab qui est plus pratique pour l’utilisateur, l’avantage de créer un plugin est que nous pouvons tirer parti de l’infrastructure de DataLab, si nous encapsulons notre fonction de traitement d’une certaine manière (voir ci-dessous) :

- Notre fonction sera exécutée dans un processus séparé, nous pouvons donc l’interrompre si elle prend trop de temps à s’exécuter.
- Les avertissements et les erreurs seront gérés par DataLab, nous n’avons donc pas besoin de les gérer nous-mêmes.

Le changement le plus significatif est que nous devons définir une fonction qui fonctionnera sur les objets d’image natifs de DataLab (`cdl.obj.ImageObj`), au lieu de fonctionner sur des tableaux NumPy. Nous devons donc trouver un moyen d’appeler notre fonction personnalisée `weighted_average_denoise` avec un `cdl.obj.ImageObj` en entrée et en sortie. Pour éviter d’écrire beaucoup de code de base, nous pouvons utiliser l’enveloppe de fonction fournie par DataLab : `cdl.computation.image.Wrap11Func`.

Par ailleurs, nous devons définir une classe qui décrit notre plugin, qui doit hériter de `cdl.plugins.PluginBase` et nommer le script Python qui contient le code du plugin avec un nom qui commence par `cdl_` (par exemple `cdl_custom_func.py`), afin que DataLab puisse le découvrir au démarrage.

De plus, dans le code du plugin, nous voulons ajouter une entrée dans le menu « Plugins », afin que l’utilisateur puisse accéder à notre plugin depuis l’interface graphique.

Voici le code du plugin :

```

1 # -*- coding: utf-8 -*-
2
3 """
4 Custom denoising filter plugin
5 =====

```

(suite sur la page suivante)

(suite de la page précédente)

```

6  This is a simple example of a DataLab image processing plugin.
7
8
9  It is part of the DataLab custom function tutorial.
10
11  .. note::
12
13      This plugin is not installed by default. To install it, copy this file to
14      your DataLab plugins directory (see `DataLab documentation
15      <https://datalab-platform.com/en/features/general/plugins.html>`).
16  """
17
18  import numpy as np
19  import scipy.ndimage as spi
20
21  import cdl.computation.image as cpi
22  import cdl.obj
23  import cdl.param
24  import cdl.plugins
25
26
27  def weighted_average_denoise(data: np.ndarray) -> np.ndarray:
28      """Apply a custom denoising filter to an image.
29
30      This filter averages the pixels in a 5x5 neighborhood, but gives less weight
31      to pixels that significantly differ from the central pixel.
32      """
33
34      def filter_func(values: np.ndarray) -> float:
35          """Filter function"""
36          central_pixel = values[len(values) // 2]
37          differences = np.abs(values - central_pixel)
38          weights = np.exp(-differences / np.mean(differences))
39          return np.average(values, weights=weights)
40
41      return spi.generic_filter(data, filter_func, size=5)
42
43
44  class CustomFilters(cdl.plugins.PluginBase):
45      """DataLab Custom Filters Plugin"""
46
47      PLUGIN_INFO = cdl.plugins.PluginInfo(
48          name="My custom filters",
49          version="1.0.0",
50          description="This is an example plugin",
51      )
52
53      def create_actions(self) -> None:
54          """Create actions"""
55          acth = self.imagepanel.acthandler
56          proc = self.imagepanel.processor
57          with acth.new_menu(self.PLUGIN_INFO.name):

```

(suite sur la page suivante)

(suite de la page précédente)

```

58     for name, func in (("Weighted average denoise", weighted_average_denoise),):
59         # Wrap function to handle ``ImageObj`` objects instead of NumPy arrays
60         wrapped_func = cpi.Wrap11Func(func)
61         acth.new_action(
62             name, triggered=lambda: proc.compute_11(wrapped_func, title=name)
63         )

```

Pour le tester, nous devons ajouter le script du plugin à l'un des répertoires de plugins qui sont découverts par DataLab au démarrage (voir la section [Plugins](#) pour plus de détails, ou le [Détection de taches sur une image](#) pour un exemple).

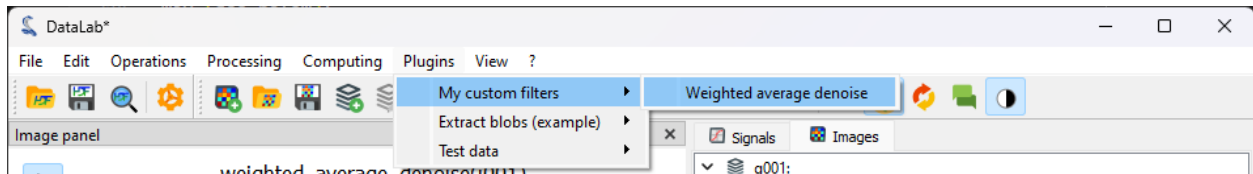


FIG. 99 – Nous redémarrons DataLab et nous pouvons voir que le plugin a été chargé.

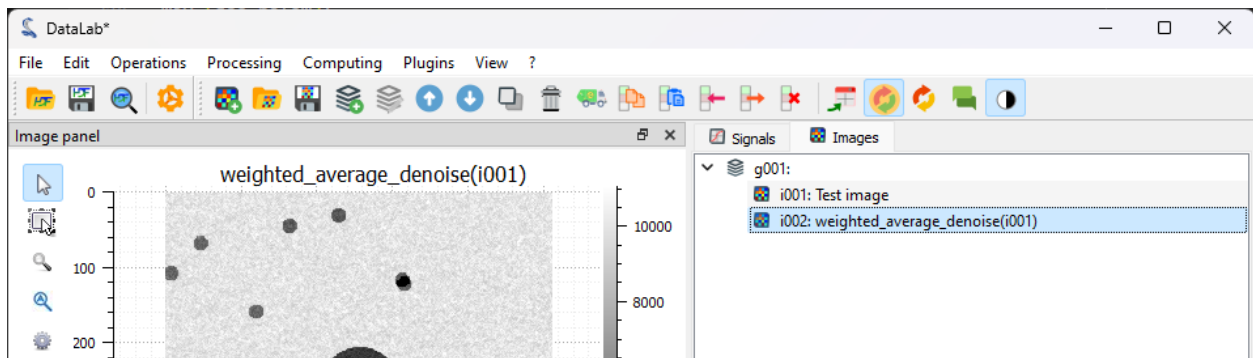
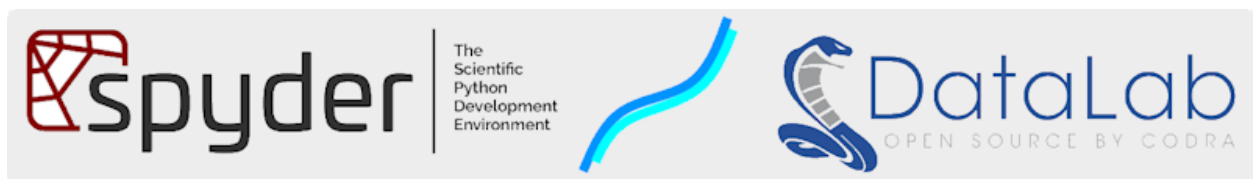


FIG. 100 – Nous générons à nouveau notre image de test en utilisant (voir les premières étapes du tutoriel), et nous la traitons en utilisant le plugin : « Plugins > My custom filters > Weighted average denoise ».

## DataLab et Spyder : un mariage parfait

Ce tutoriel montre comment utiliser [Spyder](#) pour travailler avec DataLab à travers un exemple, en utilisant des algorithmes et des données fictifs qui représentent un travail de recherche/technique hypothétique. L'objectif est d'illustrer comment utiliser DataLab pour tester vos algorithmes avec des données, et comment les déboguer si nécessaire.

L'exemple est assez simple, mais il illustre les concepts de base du travail avec DataLab *et* [Spyder](#).



**Note :** DataLab et [Spyder](#) sont des outils **complémentaires**. Alors que [Spyder](#) est un environnement de développement puissant avec des capacités de calcul scientifique interactif, DataLab est un outil d'analyse de données polyvalent qui peut être utilisé pour effectuer un large éventail de tâches, de la simple visualisation de données à l'analyse et au traitement de données complexes. En d'autres termes, [Spyder](#) est un outil de **développement**, tandis que DataLab est

un outil d'**analyse de données**. Vous pouvez utiliser [Spyder](#) pour développer des algorithmes, puis utiliser DataLab pour analyser les données avec ces algorithmes.

## Concepts de base

Dans le contexte de votre travail de recherche ou technique, nous supposons que vous développez un logiciel pour traiter des données (signaux ou images) : ce logiciel peut être une application autonome ou une bibliothèque que vous utiliserez dans d'autres applications, ou même un simple script que vous exécuterez à partir de la ligne de commande. Dans tous les cas, vous devrez suivre un processus de développement qui comprendra les étapes suivantes :

0. Prototyper l'algorithme dans un environnement de développement, tel que [Spyder](#).
1. Développer l'algorithme dans un environnement de développement, tel que [Spyder](#).
2. Tester l'algorithme avec des données.
3. Déboguer l'algorithme si nécessaire.
4. Répéter les étapes 2 et 3 jusqu'à ce que l'algorithme fonctionne comme prévu.
5. Utiliser l'algorithme dans votre application.

**Note :** DataLab peut vous aider avec l'étape 0 car il fournit toutes les primitives de traitement dont vous avez besoin pour prototyper votre algorithme : vous pouvez charger des données, les visualiser et effectuer des opérations de traitement de base. Nous ne couvrirons pas cette étape dans les paragraphes suivants car la documentation de DataLab fournit déjà beaucoup d'informations à ce sujet.

Dans ce tutoriel, nous verrons comment utiliser DataLab pour effectuer les étapes 2 et 3. Nous supposons que vous avez déjà prototypé (de préférence dans DataLab !) et développé votre algorithme dans [Spyder](#). Maintenant, vous voulez le tester avec des données, mais sans quitter [Spyder](#) car vous devrez peut-être apporter des modifications à votre algorithme et le re-tester. De plus, votre flux de travail est déjà configuré dans [Spyder](#) et vous ne voulez pas le changer.

**Note :** Dans ce tutoriel, nous supposons que vous avez déjà installé DataLab et que vous l'avez démarré. Si vous ne l'avez pas encore fait, veuillez vous référer à la section [Installation](#) de la documentation.

De plus, nous supposons que vous avez déjà installé [Spyder](#) et que vous l'avez démarré. Si vous ne l'avez pas encore fait, veuillez vous référer à la documentation de [Spyder](#). **Notez que vous n'avez pas besoin d'installer DataLab dans le même environnement que Spyder.** : c'est tout l'intérêt de DataLab, c'est une application autonome qui peut être utilisée à partir de n'importe quel environnement. Pour ce tutoriel, vous n'avez besoin d'installer que le client DataLab Simple (*pip install cdlclient*) dans le même environnement que [Spyder](#).

## Tester votre algorithme avec DataLab

Supposons que vous ayez développé des algorithmes dans le module `my_work` de votre projet. Vous les avez déjà prototypés dans DataLab, et vous les avez développés dans [Spyder](#) en écrivant des fonctions qui prennent des données en entrée et renvoient des données traitées en sortie. Maintenant, vous voulez tester ces algorithmes avec des données.

Pour tester ces algorithmes, vous avez écrit deux fonctions dans le module `my_work` :

- `test_my_1d_algorithm` : cette fonction renvoie des données 1D qui vous permettront de valider votre premier algorithme qui fonctionne sur des données 1D.
- `test_my_2d_algorithm` : cette fonction renvoie des données 2D qui vous permettront de valider votre deuxième algorithme qui fonctionne sur des données 2D.

Vous pouvez maintenant utiliser DataLab pour visualiser les données renvoyées par ces fonctions directement depuis [Spyder](#) :

- Tout d'abord, vous devez démarrer DataLab et [Spyder](#).

- Rappelez-vous que DataLab est une application autonome qui peut être utilisée à partir de n'importe quel environnement, vous n'avez donc pas besoin de l'installer dans le même environnement que [Spyder](#) car la connexion entre ces deux applications se fait via un protocole de communication.

Voici comment faire :

```
# %% Connecting to DataLab current session

from cdlclient import SimpleRemoteProxy

proxy = SimpleRemoteProxy()
proxy.connect()

# %% Visualizing 1D data from my work

from my_work import test_my_1d_algorithm

x, y = test_my_1d_algorithm() # Here is all my research/technical work!
proxy.add_signal("My 1D result data", x, y) # Let's visualize it in DataLab
proxy.compute_wiener() # Denoise the signal using the Wiener filter

# %% Visualizing 2D data from my work

from my_work import test_my_2d_algorithm

z = test_my_2d_algorithm()[1] # Here is all my research/technical work!
proxy.add_image("My 2D result data", z) # Let's visualize it in DataLab
```

Si nous exécutons les deux premières cellules, nous verrons la sortie suivante dans la console [Spyder](#) :

```
Python 3.11.5 (tags/v3.11.5:cce6ba9, Aug 24 2023, 14:38:34) [MSC v.1936 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.15.0 -- An enhanced Interactive Python.

In [1]: runcell('Connecting to DataLab current session', 'my_work_test_with_cdl.py')
Connecting to DataLab XML-RPC server...OK (port: 54577)

In [2]: runcell('Visualizing 1D data from my work', 'my_work_test_with_cdl.py')
Out[2]: True
```

## Déboguer votre algorithme avec DataLab

Maintenant que vous avez testé vos algorithmes avec des données, vous voudrez peut-être les déboguer si nécessaire. Pour ce faire, vous pouvez combiner les capacités de débogage de [Spyder](#) avec DataLab.

Voici le code de l'algorithme fictif que nous voulons déboguer, dans lequel nous avons introduit un paramètre optionnel `debug_with_datalab` qui - s'il est défini sur `True` - créera un objet proxy permettant de visualiser les données étape par étape dans DataLab :

```
def generate_2d_data(
    num_lines: int = 10,
    noise_std: float = 0.1,
```

(suite sur la page suivante)

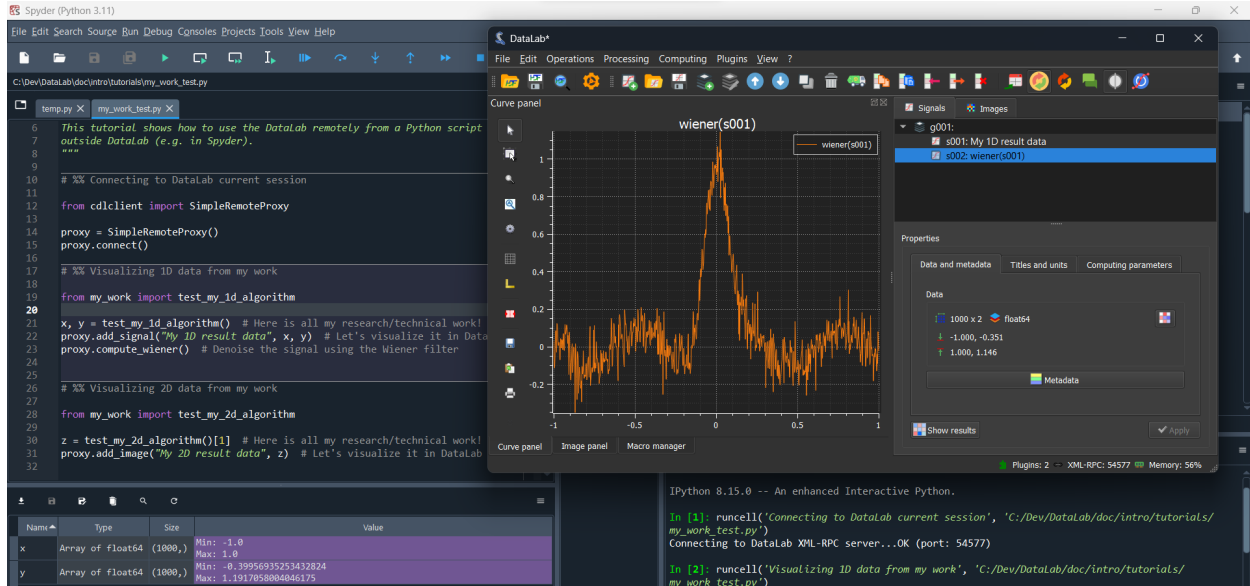


FIG. 101 – Sur cette capture d’écran, nous pouvons voir le résultat de l’évaluation des deux premières cellules : la première cellule se connecte à DataLab, et la deuxième cellule visualise les données 1D renvoyées par la fonction `test_my_1d_algorithm`.

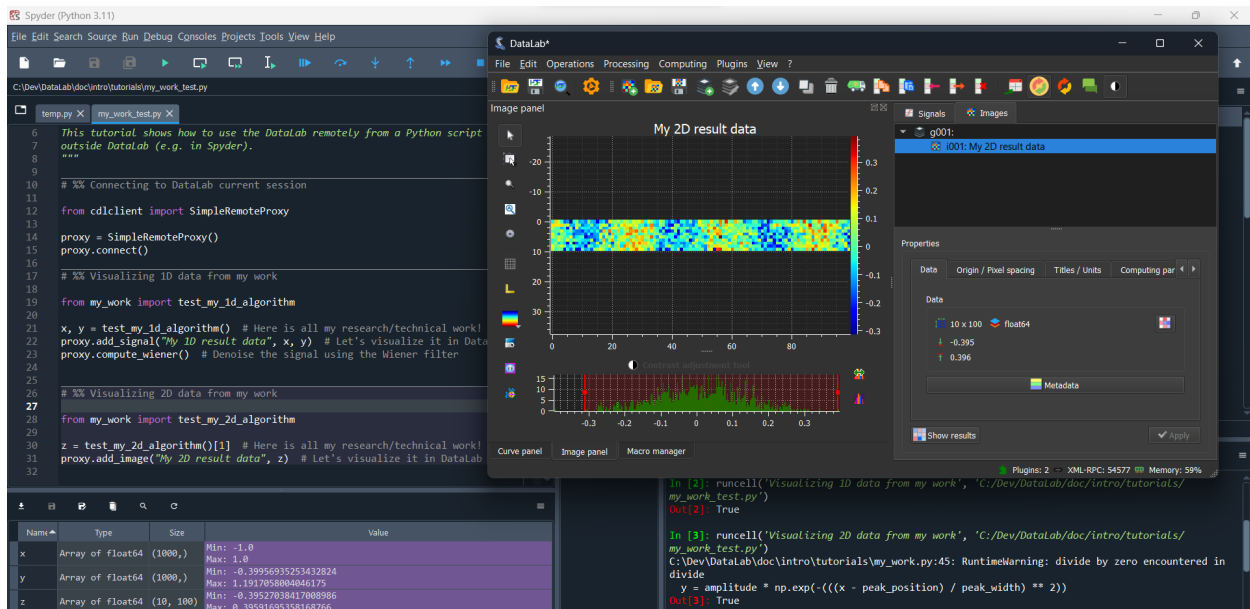


FIG. 102 – Sur cette capture d’écran, nous pouvons voir le résultat de l’évaluation de la troisième cellule : la fonction `test_my_2d_algorithm` renvoie un tableau 2D, et nous pouvons le visualiser directement dans DataLab.

(suite de la page précédente)

```

x_min: float = -1,
x_max: float = 1,
num_points: int = 100,
debug_with_datalab: bool = False,
) -> tuple[np.ndarray, np.ndarray]:
    """Generate 2D data that can be used in the tutorials to represent some results.

    Args:
        num_lines: Number of lines in the generated data, by default 10.
        noise_std: Standard deviation of the noise, by default 0.1.
        x_min: Minimum value of the x axis, by default -1.
        x_max: Maximum value of the x axis, by default 1.
        num_points: Number of points in the generated data, by default 100.
        debug_with_datalab: Whether to use the DataLab to debug the function,
            by default False.

    Returns:
        Generated data (x, y; where y is a 2D array and x is a 1D array).
    """
    proxy = None
    if debug_with_datalab:
        proxy = SimpleRemoteProxy()
        proxy.connect()
    z = np.zeros((num_lines, num_points))
    for i in range(num_lines):
        amplitude = 0.1 * i**2
        peak_position = 0.5 * i**2
        peak_width = 0.1 * i**2
        x, y = generate_1d_data(
            amplitude,
            peak_position,
            peak_width,
            relaxation_oscillations=True,
            noise=True,
            noise_std=noise_std,
            x_min=x_min,
            x_max=x_max,
            num_points=num_points,
        )
        z[i] = y
        if proxy is not None:
            proxy.add_signal(f"Line {i}", x, y)
    return x, z

```

La fonction `test_my_2d_algorithm` correspondante a également un paramètre optionnel `debug_with_datalab` qui est simplement passé à la fonction `generate_2d_data`.

Maintenant, nous pouvons utiliser `Spyder` pour déboguer la fonction `test_my_2d_algorithm` :

```

# %% Debugging my work with DataLab

from my_work import generate_2d_data

```

(suite sur la page suivante)

(suite de la page précédente)

```
x, z = generate_2d_data(debug_with_datalab=True)
```

Dans cet exemple simple, l'algorithme itère simplement 10 fois et génère un tableau 1D à chaque itération. Chaque tableau 1D est ensuite empilé dans un tableau 2D qui est renvoyé par la fonction `generate_2d_data`. Avec le paramètre `debug_with_datalab` défini sur `True`, nous pouvons visualiser chaque tableau 1D dans DataLab : de cette façon, nous pouvons vérifier que l'algorithme fonctionne comme prévu.

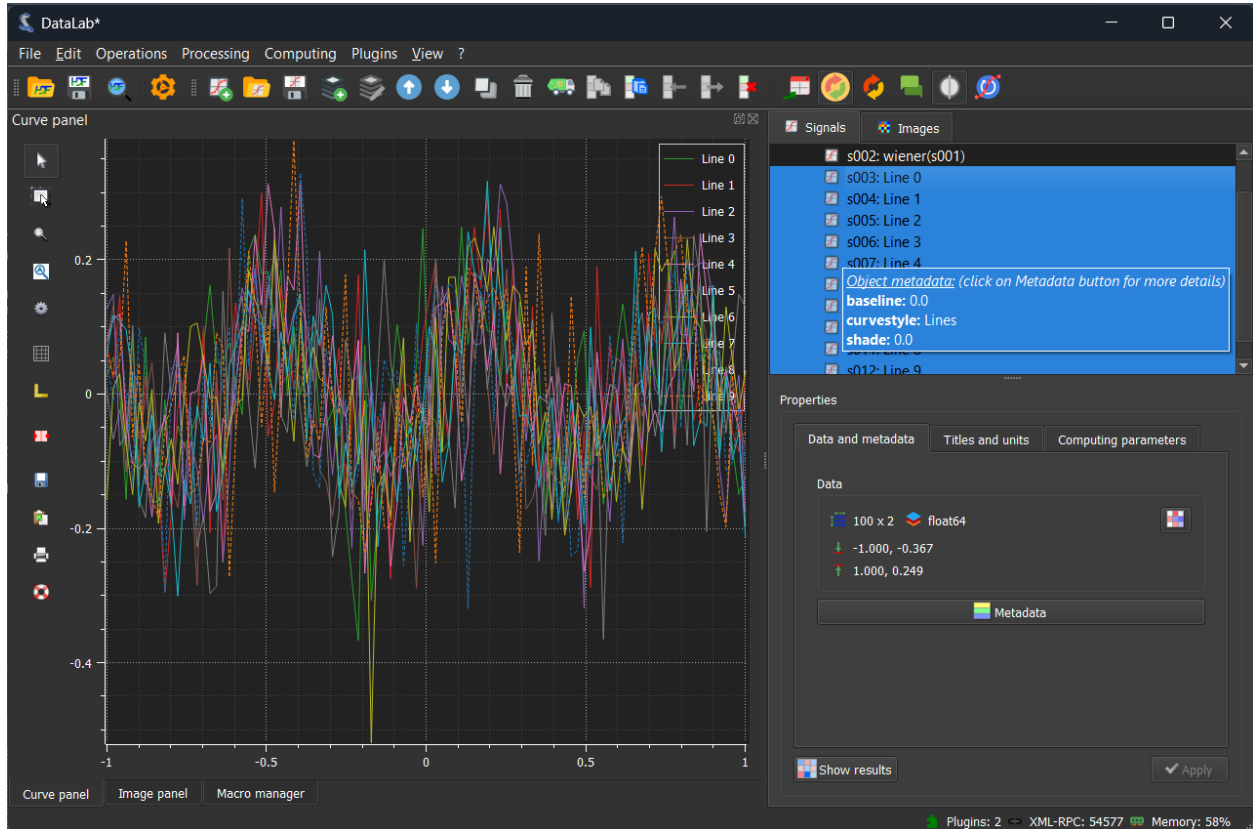


Fig. 103 – Sur cette capture d'écran, nous pouvons voir le résultat de l'évaluation de la première cellule : la fonction `test_my_2d_algorithm` est appelée avec le paramètre `debug_with_datalab` défini sur `True` : 10 tableaux 1D sont générés et visualisés dans DataLab.

**Note :** Si nous avions exécuté le script en utilisant le débogueur `Spyder` et défini un point d'arrêt dans la fonction `generate_2d_data`, nous aurions vu les tableaux 1D générés dans DataLab à chaque itération : comme DataLab est exécuté dans un processus séparé, nous aurions pu manipuler les données dans DataLab pendant que l'algorithme est en pause dans `Spyder`.



# CHAPITRE 2

---

## Fonctionnalités

---

Les paragraphes suivants décrivent les fonctionnalités de DataLab, la plateforme open-source d'analyse et de visualisation de données scientifiques.

Tout d'abord, le paragraphe **validation** explique pourquoi et comment DataLab est validé - c'est une autre fonctionnalité singulière et fondamentale de DataLab.

Ensuite, le paragraphe **fonctionnalités générales** décrit les fonctionnalités générales de DataLab, qui concernent à la fois les panneaux de traitement du signal et d'image.

Les paragraphes **traitement du signal** et **traitement d'image** décrivent les fonctionnalités spécifiques aux panneaux de traitement du signal et d'image, respectivement.

---

**Note :** Avant de plonger dans les détails des autres parties de la documentation, il est recommandé de commencer par la page [Espace de travail](#), qui décrit les concepts de base de DataLab.

---

### Voir aussi :

Pour une vue d'ensemble synthétique des fonctionnalités de DataLab, veuillez vous référer à la page [Fonctionnalités principales](#).

## 2.1 Validation

DataLab est une plateforme d'analyse et de visualisation de données scientifiques qui peut être utilisée dans diverses disciplines scientifiques, y compris la biologie, la physique et l'astronomie. Le point commun de toutes ces disciplines est la nécessité de valider les résultats de l'analyse informatique par rapport à des données de référence. Il s'agit d'une étape critique dans la méthode scientifique et elle est essentielle pour la reproductibilité et la confiance dans les résultats. C'est ce que nous appelons la **validation technique**.

DataLab est également utilisé dans des applications industrielles, où la validation des résultats est essentielle pour garantir la qualité du processus, mais une gamme plus large de validation est nécessaire pour s'assurer qu'un maximum de cas d'utilisation sont couverts d'un point de vue fonctionnel. C'est ce que nous appelons la **validation fonctionnelle**.

Ainsi, la validation de DataLab peut être classée en deux types :

- La **validation technique** : garantit que les résultats de l'analyse informatique sont précis et fiables.
- La **validation fonctionnelle** : vérifie que le logiciel se comporte comme prévu dans une variété de cas d'utilisation (suite de tests unitaires automatisés classiques).

### 2.1.1 Validation technique

La validation technique de DataLab est basée sur deux concepts clés : les **données de référence** et la **validation analytique**.

#### Données de référence

Les données de référence sont des données connues pour être correctes. Elles sont utilisées pour valider les résultats de l'analyse informatique.

Dans DataLab, les données de référence peuvent être obtenues à partir de diverses sources, y compris :

- Données expérimentales
- Données simulées
- Données synthétiques
- Données provenant d'une source de confiance

#### Validation analytique

La validation analytique est le processus de comparaison des résultats de l'analyse informatique avec les données de référence. Cela est fait pour s'assurer que les résultats sont précis et fiables.

Dans DataLab, la validation analytique est mise en œuvre à l'aide de diverses techniques, y compris :

- Validation croisée avec un modèle analytique (provenant d'une source de confiance, par exemple [SciPy](#) ou [NumPy](#))
- Analyse statistique
- Inspection visuelle
- Examen par un expert

#### Périmètre

Le périmètre de la validation technique dans DataLab inclut toutes les fonctions de calcul qui opèrent sur les objets de signal et d'image de DataLab (c'est-à-dire [cdl.obj.SignalObj](#) et [cdl.obj.ImageObj](#)).

Cela inclut des fonctions pour (toutes les fonctions sont nommées `compute_<function_name>`) :

- Le traitement du signal ([cdl.computation.signal](#))
- Le traitement d'image ([cdl.computation.image](#))

#### Implémentation

Les tests sont implémentés en utilisant le cadre [pytest](#).

Lors de l'écriture d'un nouveau test de validation technique, les règles suivantes doivent être suivies concernant la fonction de test :

- La fonction de test doit être nommée :
  - `test_signal_<function_name>` pour les fonctions de calcul de signal
  - `test_image_<function_name>` pour les fonctions de calcul d'image

---

**Note :** Le préfixe `signal` ou `image` est utilisé pour indiquer le type d'objet sur lequel la fonction opère. Il peut être omis si la fonction opère exclusivement sur un type d'objet (par exemple, `test_adjust_gamma` est la fonction de test pour la fonction `compute_adjust_gamma`, qui opère sur des images).

---

- La fonction de test doit être marquée avec le décorateur `@pytest.mark.validation`.

En suivant ces règles, on s'assure que :

- Les tests sont facilement identifiés comme des tests de validation technique.
- Les tests peuvent être exécutés séparément à l'aide de l'interface en ligne de commande (voir [Exécuter les tests de validation technique](#)).
- Les tests sont automatiquement découverts pour synthétiser l'état de validation des fonctions de calcul (voir [Etat de validation de DataLab](#)).

## Exécution des tests

Dans DataLab, les tests de validation technique sont disséminés dans la suite de tests du projet, mais ils peuvent également être exécutés séparément à l'aide de l'interface en ligne de commande.

### Voir aussi :

Voir le paragraphe [Exécuter les tests de validation technique](#) pour plus d'informations sur la façon d'exécuter les tests de validation technique.

## 2.1.2 Validation fonctionnelle

### Stratégie

La validation fonctionnelle de DataLab est basée sur une stratégie de test classique, avec un fort accent sur les tests automatisés. À l'exception d'un ou deux tests manuels (par exemple, un test de charge), tous les tests sont automatisés (plus de 99% des tests sont automatisés).

L'écriture des tests suit le principe TDD (Test-Driven Development) :

- Lorsqu'une *nouvelle fonctionnalité* est développée, le développeur écrit d'abord les tests. Les tests sont ensuite exécutés pour s'assurer qu'ils échouent. Le développeur implémente ensuite la fonctionnalité, et les tests sont exécutés à nouveau pour s'assurer qu'ils réussissent.
- Lorsqu'un *bug est signalé*, le développeur écrit un test qui reproduit le bug. Le test est exécuté pour s'assurer qu'il échoue. Le développeur corrige ensuite le bug, et le test est exécuté à nouveau pour s'assurer qu'il réussit.

Selon le niveau d'abstraction, des tests unitaires et/ou des tests d'application sont écrits. Lors de l'écriture des deux types de tests, le développeur commence par les tests unitaires puis écrit les tests d'application.

### Types de tests

La validation fonctionnelle de DataLab repose sur deux types principaux de tests :

- Les **tests unitaires** (scripts de test nommés `*_unit_test.py`) : testent les fonctions ou méthodes individuelles. Tous les tests unitaires sont automatisés.
- Les **tests applicatifs** (scripts de test nommés `*_app_test.py`) : testent l'interaction entre les composants (tests d'intégration), ou l'application dans son ensemble. Tous les tests applicatifs sont automatisés.

## Implémentation

Les tests sont implémentés en utilisant le framework `pytest`. De nombreux tests existants peuvent être dérivés pour créer de nouveaux tests.

### Exécution des tests

Pour exécuter les tests, le développeur utilise l'interface en ligne de commande. Voir la section *Exécuter l'ensemble des tests* pour plus d'informations sur la façon d'exécuter les tests de validation fonctionnelle.

## 2.1.3 Etat de validation de DataLab

### Validation fonctionnelle

Dans DataLab, la validation fonctionnelle est basée sur une stratégie de test classique (voir *Validation fonctionnelle*).

La couverture de test est d'environ 90%, avec plus de 200 tests.

### Validation technique

Ce paragraphe fournit l'état de validation des fonctions de calcul dans DataLab (c'est ce que nous appelons validation technique, voir *Validation technique*).

---

**Note :** Il s'agit d'un travail en cours : les tableaux ci-dessous sont mis à jour en continu à mesure que de nouvelles fonctions sont validées ou que le code de test est adapté (les tableaux sont générés à partir du code de test). Certaines fonctions sont déjà validées mais n'apparaissent pas encore dans la liste ci-dessous, tandis que d'autres sont encore en cours de validation.

---

**Avertissement :** L'état de validation ne doit pas être confondu avec la couverture de test. L'état de validation indique si la fonction a été validée par rapport à des données de référence ou des modèles analytiques. La couverture de test indique le pourcentage du code qui est exécuté par la suite de tests, mais elle ne prend pas nécessairement en compte la justesse des résultats (la couverture de test de DataLab est d'environ 90%).

TABLEAU 1 – Statistiques de validation

Catégorie	Signal	Image	Total
Nombre de fonctions de calcul	51	94	145
Nombre de fonctions de calcul validées	44	82	126
Pourcentage de fonctions de calcul validées	86%	87%	86%

## Fonctions de calcul signal

Le tableau ci-dessous montre l'état de validation des fonctions de calcul signal dans DataLab. Il est généré automatiquement à partir du code source.

TABLEAU 2: Etat de validation de DataLab

Fonctions de calcul	Description	Fonction de test
<code>compute_abs</code>	Compute absolute value with <code>numpy.absolute</code>	<code>test_signal_abs</code>
<code>compute_addition</code>	Add <b>dst</b> and <b>src</b> signals and return <b>dst</b> signal modified in place	<code>test_signal_addition</code>
<code>compute_addition_constant</code>	Add <b>dst</b> and a constant value and return a the new result signal object	<code>test_signal_addition_constant</code>
<code>compute_arithmetic</code>	Perform arithmetic operation on two signals	<code>test_signal_arithmetic</code>
<code>compute_astype</code>	Convert data type with <code>numpy.astype()</code>	<code>test_signal_astype</code>
<code>compute_bandwidth_3db</code>	Compute bandwidth at -3 dB with <code>cdl.algorithms.signal.bandwidth()</code>	<code>test_signal_bandwidth_3db</code>
<code>compute_calibration</code>	Compute linear calibration	<code>test_signal_calibration</code>
<code>compute_clip</code>	Compute maximum data clipping with <code>numpy.clip()</code>	<code>test_signal_clip</code>
<code>compute_contrast</code>	Compute contrast with <code>cdl.algorithms.signal.contrast()</code>	<code>test_signal_contrast</code>
<code>compute_convolution</code>	Compute convolution of two signals	<code>test_signal_convolution</code>
<code>compute_derivative</code>	Compute derivative with <code>numpy.gradient()</code>	<code>test_signal_derivative</code>
<code>compute_detrending</code>	Detrend data with <code>scipy.signal.detrend()</code>	N/A
<code>compute_difference</code>	Compute difference between two signals	<code>test_signal_difference</code>
<code>compute_difference_constant</code>	Subtract a constant value from a signal	<code>test_signal_difference_constant</code>
<code>compute_division</code>	Compute division between two signals	<code>test_signal_division</code>
<code>compute_division_constant</code>	Divide a signal by a constant value	<code>test_signal_division_constant</code>
<code>compute_dynamic_parameters</code>	Compute Dynamic parameters	<code>test_dynamic_parameters</code>
<code>compute_exp</code>	Compute exponential with <code>numpy.exp</code>	<code>test_signal_exp</code>
<code>compute_fft</code>	Compute FFT with <code>cdl.algorithms.signal.fft1d()</code>	<code>test_signal_fft</code>
<code>compute_filter</code>	Compute frequency filter (low-pass, high-pass, band-pass, band-stop)	N/A
<code>compute_fwle2</code>	Compute FW at $1/e^2$ with <code>cdl.algorithms.signal.fwle2()</code>	<code>test_signal_fwle2</code>
<code>compute_fwhm</code>	Compute FWHM with <code>cdl.algorithms.signal.fwhm()</code>	<code>test_signal_fwhm</code>
<code>compute_gaussian_filter</code>	Compute gaussian filter with <code>scipy.ndimage.gaussian_filter()</code>	<code>test_signal_gaussian_filter</code>
<code>compute_histogram</code>	Compute histogram with <code>numpy.histogram()</code>	N/A
<code>compute_iftt</code>	Compute iFFT with <code>cdl.algorithms.signal.iftt1d()</code>	<code>test_signal_iftt</code>
<code>compute_im</code>	Compute imaginary part with <code>numpy.imag()</code>	<code>test_signal_im</code>
<code>compute_integral</code>	Compute integral with <code>scipy.integrate.cumulative_trapezoid()</code>	<code>test_signal_integral</code>

suite sur la page suivante

Tableau 2 – suite de la page précédente

Fonctions de calcul	Description	Fonction de test
<code>compute_interpolation</code>	Interpolate data with <code>cdl.algorithms.signal.interpolate()</code>	N/A
<code>compute_log10</code>	Compute Log10 with <code>numpy.log10</code>	<code>test_signal_log10</code>
<code>compute_magnitude_spectrum</code>	Compute magnitude spectrum	<code>test_signal_magnitude_spectrum</code>
<code>compute_moving_average</code>	Compute moving average with <code>scipy.ndimage.uniform_filter()</code>	<code>test_signal_moving_average</code>
<code>compute_moving_median</code>	Compute moving median with <code>scipy.ndimage.median_filter()</code>	<code>test_signal_moving_median</code>
<code>compute_normalize</code>	Normalize data with <code>cdl.algorithms.signal.normalize()</code>	<code>test_signal_normalize</code>
<code>compute_offset_correction</code>	Correct offset : subtract the mean value of the signal in the specified range	<code>test_signal_offset_correction</code>
<code>compute_peak_detection</code>	Peak detection with <code>cdl.algorithms.signal.peak_indices()</code>	N/A
<code>compute_phase_spectrum</code>	Compute phase spectrum	<code>test_signal_phase_spectrum</code>
<code>compute_power</code>	Compute power with <code>numpy.power</code>	<code>test_signal_power</code>
<code>compute_product</code>	Multiply <b>dst</b> and <b>src</b> signals and return <b>dst</b> signal modified in place	<code>test_signal_product</code>
<code>compute_product_constant</code>	Multiply <b>dst</b> by a constant value and return the new result signal object	<code>test_signal_product_constant</code>
<code>compute_psd</code>	Compute power spectral density	<code>test_signal_psd</code>
<code>compute_quadratic_differer</code>	Compute quadratic difference between two signals	<code>test_quadratic_difference</code>
<code>compute_re</code>	Compute real part with <code>numpy.real()</code>	<code>test_signal_re</code>
<code>compute_resampling</code>	Resample data with <code>cdl.algorithms.signal.interpolate()</code>	N/A
<code>compute_reverse_x</code>	Reverse x-axis	<code>test_signal_reverse_x</code>
<code>compute_sampling_rate_peri</code>	Compute sampling rate and period	<code>test_signal_sampling_rate_period</code>
<code>compute_sqrt</code>	Compute square root with <code>numpy.sqrt</code>	<code>test_signal_sqrt</code>
<code>compute_stats</code>	Compute statistics on a signal	<code>test_signal_stats_unit</code>
<code>compute_swap_axes</code>	Swap axes	<code>test_signal_swap_axes</code>
<code>compute_wiener</code>	Compute Wiener filter with <code>scipy.signal.wiener()</code>	<code>test_signal_wiener</code>
<code>compute_windowing</code>	Compute windowing (available methods : hamming, hanning, bartlett, blackman)	N/A
<code>compute_x_at_minmax</code>	Compute x at min/max	<code>test_signal_x_at_minmax</code>

## Fonctions de calcul image

Le tableau ci-dessous montre l'état de validation des fonctions de calcul image dans DataLab. Il est généré automatiquement à partir du code source.

TABLEAU 3: Etat de validation des fonctions de calcul image

Fonctions de calcul	Description	Fonction de test
<code>compute_abs</code>	Compute absolute value with <code>numpy.absolute</code>	<code>test_image_abs</code>
<code>compute_addition</code>	Add <b>dst</b> and <b>src</b> images and return <b>dst</b> image modified in place	<code>test_image_addition</code>

suite sur la page suivante

Tableau 3 – suite de la page précédente

Fonctions de calcul	Description	Fonction de test
<code>compute_addition_constant</code>	Add <code>dst</code> and a constant value and return the new result image object	<code>test_image_addition_constant</code>
<code>compute_arithmetic</code>	Compute arithmetic operation on two images	<code>test_image_arithmetic</code>
<code>compute_astype</code>	Convert image data type with <code>cdl.algorithms.datatypes.clip_astype()</code>	<code>test_image_astype</code>
<code>compute_average_profile</code>	Compute horizontal or vertical average profile	<code>test_average_profile</code>
<code>compute_binning</code>	Binning function on data with <code>cdl.algorithms.image.binning()</code>	<code>test_binning</code>
<code>compute_butterworth</code>	Compute Butterworth filter with <code>skimage.filters.butterworth()</code>	<code>test_butterworth</code>
<code>compute_calibration</code>	Compute linear calibration	<code>test_image_calibration</code>
<code>compute_centroid</code>	Compute centroid	<code>test_image_centroid</code>
<code>compute_clip</code>	Apply clipping with <code>numpy.clip()</code>	<code>test_image_clip</code>
<code>compute_difference</code>	Compute difference between two images	<code>test_image_difference</code>
<code>compute_difference_constant</code>	Subtract a constant value from an image and return the new result image object	<code>test_image_difference_constant</code>
<code>compute_division</code>	Compute division between two images	<code>test_image_division</code>
<code>compute_division_constant</code>	Divide an image by a constant value and return the new result image object	<code>test_image_division_constant</code>
<code>compute_enclosing_circle</code>	Compute minimum enclosing circle	N/A
<code>compute_exp</code>	Compute exponential with <code>numpy.exp</code>	<code>test_image_exp</code>
<code>compute_fft</code>	Compute FFT with <code>cdl.algorithms.image.fft2d()</code>	<code>test_image_fft</code>
<code>compute_flatfield</code>	Compute flat field correction with <code>cdl.algorithms.image.flatfield()</code>	N/A
<code>compute_fliph</code>	Flip data horizontally with <code>numpy.fliplr()</code>	<code>test_image_fliph</code>
<code>compute_flipv</code>	Flip data vertically with <code>numpy.flipud()</code>	<code>test_image_flipv</code>
<code>compute_gaussian_filter</code>	Compute gaussian filter with <code>scipy.ndimage.gaussian_filter()</code>	<code>test_image_gaussian_filter</code>
<code>compute_histogram</code>	Compute histogram of the image data, with <code>numpy.histogram()</code>	N/A
<code>compute_hough_circle_peaks</code>	Compute Hough circles	N/A
<code>compute_iff</code>	Compute inverse FFT with <code>cdl.algorithms.image.iff2d()</code>	<code>test_image_iff</code>
<code>compute_im</code>	Compute imaginary part with <code>numpy.imag()</code>	<code>test_image_im</code>
<code>compute_line_profile</code>	Compute horizontal or vertical profile	<code>test_line_profile</code>
<code>compute_log10</code>	Compute log10 with <code>numpy.log10</code>	<code>test_image_log10</code>
<code>compute_logp1</code>	Compute log10(z+n) with <code>numpy.log10</code>	<code>test_image_logp1</code>
<code>compute_magnitude_spectrum</code>	Compute magnitude spectrum	<code>test_image_magnitude_spectrum</code>
<code>compute_moving_average</code>	Compute moving average with <code>scipy.ndimage.uniform_filter()</code>	<code>test_image_moving_average</code>
<code>compute_moving_median</code>	Compute moving median with <code>scipy.ndimage.median_filter()</code>	<code>test_image_moving_median</code>
<code>compute_normalize</code>		<code>test_image_normalize</code>
<code>compute_offset_correction</code>	Apply offset correction	<code>test_image_offset_correction</code>

suite sur la page suivante

Tableau 3 – suite de la page précédente

Fonctions de calcul	Description	Fonction de test
<code>compute_phase_spectrum</code>	Compute phase spectrum	<code>test_image_phase_spectrum</code>
<code>compute_product</code>	Multiply <b>dst</b> and <b>src</b> images and return <b>dst</b> image modified in place	<code>test_image_product</code>
<code>compute_product_constant</code>	Multiply <b>dst</b> by a constant value and return the new result image object	<code>test_image_product_constant</code>
<code>compute_psd</code>	Compute power spectral density	<code>test_image_psd</code>
<code>compute_quadratic_differer</code>	Compute quadratic difference between two images	<code>test_quadratic_difference</code>
<code>compute_radial_profile</code>	Compute radial profile around the centroid	N/A
<code>compute_re</code>	Compute real part with <code>numpy.real()</code>	<code>test_image_re</code>
<code>compute_resize</code>	Zooming function with <code>scipy.ndimage.zoom()</code>	N/A
<code>compute_rotate</code>	Rotate data with <code>scipy.ndimage.rotate()</code>	<code>test_image_rotate</code>
<code>compute_rotate270</code>	Rotate data 270° with <code>numpy.rot90()</code>	<code>test_image_rotate270</code>
<code>compute_rotate90</code>	Rotate data 90° with <code>numpy.rot90()</code>	<code>test_image_rotate90</code>
<code>compute_segment_profile</code>	Compute segment profile	<code>test_segment_profile</code>
<code>compute_stats</code>	Compute statistics on an image	<code>test_image_stats_unit</code>
<code>compute_swap_axes</code>	Swap image axes with <code>numpy.transpose()</code>	<code>test_image_swap_axes</code>
<code>compute_wiener</code>	Compute Wiener filter with <code>scipy.signal.wiener()</code>	<code>test_image_wiener</code>
<code>compute_blob_dog</code>	Compute blobs using Difference of Gaussian method	N/A
<code>compute_blob_doh</code>	Compute blobs using Determinant of Hessian method	N/A
<code>compute_blob_log</code>	Compute blobs using Laplacian of Gaussian method	N/A
<code>compute_blob_opencv</code>	Compute blobs using OpenCV	N/A
<code>compute_contour_shape</code>	Compute contour shape fit	N/A
<code>compute_peak_detection</code>	Compute 2D peak detection	N/A
<code>compute_canny</code>	Compute Canny filter with <code>skimage.feature.canny()</code>	<code>test_canny</code>
<code>compute_farid</code>	Compute Farid filter with <code>skimage.filters.farid()</code>	<code>test_farid</code>
<code>compute_farid_h</code>	Compute horizontal Farid filter with <code>skimage.filters.farid_h()</code>	<code>test_farid_h</code>
<code>compute_farid_v</code>	Compute vertical Farid filter with <code>skimage.filters.farid_v()</code>	<code>test_farid_v</code>
<code>compute_laplace</code>	Compute Laplace filter with <code>skimage.filters.laplace()</code>	<code>test_laplace</code>
<code>compute_prewitt</code>	Compute Prewitt filter with <code>skimage.filters.prewitt()</code>	<code>test_prewitt</code>
<code>compute_prewitt_h</code>	Compute horizontal Prewitt filter with <code>skimage.filters.prewitt_h()</code>	<code>test_prewitt_h</code>
<code>compute_prewitt_v</code>	Compute vertical Prewitt filter with <code>skimage.filters.prewitt_v()</code>	<code>test_prewitt_v</code>
<code>compute_roberts</code>	Compute Roberts filter with <code>skimage.filters.roberts()</code>	<code>test_roberts</code>

suite sur la page suivante

Tableau 3 – suite de la page précédente

Fonctions de calcul	Description	Fonction de test
<code>compute_scharr</code>	Compute Scharr filter with <code>skimage.filters.scharr()</code>	<code>test_scharr</code>
<code>compute_scharr_h</code>	Compute horizontal Scharr filter with <code>skimage.filters.scharr_h()</code>	<code>test_scharr_h</code>
<code>compute_scharr_v</code>	Compute vertical Scharr filter with <code>skimage.filters.scharr_v()</code>	<code>test_scharr_v</code>
<code>compute_sobel</code>	Compute Sobel filter with <code>skimage.filters.sobel()</code>	<code>test_sobel</code>
<code>compute_sobel_h</code>	Compute horizontal Sobel filter with <code>skimage.filters.sobel_h()</code>	<code>test_sobel_h</code>
<code>compute_sobel_v</code>	Compute vertical Sobel filter with <code>skimage.filters.sobel_v()</code>	<code>test_sobel_v</code>
<code>compute_adjust_gamma</code>	Gamma correction with <code>skimage.exposure.adjust_gamma()</code>	<code>test_adjust_gamma</code>
<code>compute_adjust_log</code>	Compute log correction with <code>skimage.exposure.adjust_log()</code>	<code>test_adjust_log</code>
<code>compute_adjust_sigmoid</code>	Compute sigmoid correction with <code>skimage.exposure.adjust_sigmoid()</code>	<code>test_adjust_sigmoid</code>
<code>compute_equalize_adapthist</code>	Adaptive histogram equalization	<code>test_equalize_adapthist</code>
<code>compute_equalize_hist</code>	Histogram equalization with <code>skimage.exposure.equalize_hist()</code>	<code>test_equalize_hist</code>
<code>compute_rescale_intensity</code>	Rescale image intensity levels	<code>test_rescale_intensity</code>
<code>compute_black_tophat</code>	Compute Black Top-Hat with <code>skimage.morphology.black_tophat()</code>	<code>test_black_tophat</code>
<code>compute_closing</code>	Compute morphological closing with <code>skimage.morphology.closing()</code>	<code>test_closing</code>
<code>compute_dilation</code>	Compute Dilation with <code>skimage.morphology.dilation()</code>	<code>test_dilation</code>
<code>compute_erosion</code>	Compute Erosion with <code>skimage.morphology.erosion()</code>	<code>test_erosion</code>
<code>compute_opening</code>	Compute morphological opening with <code>skimage.morphology.opening()</code>	<code>test_opening</code>
<code>compute_white_tophat</code>	Compute White Top-Hat with <code>skimage.morphology.white_tophat()</code>	<code>test_white_tophat</code>
<code>compute_denoise_bilateral</code>	Compute bilateral filter denoising	<code>test_denoise_bilateral</code>
<code>compute_denoise_tophat</code>	Denoise using White Top-Hat	<code>test_denoise_tophat</code>
<code>compute_denoise_tv</code>	Compute Total Variation denoising	<code>test_denoise_tv</code>
<code>compute_denoise_wavelet</code>	Compute Wavelet denoising	<code>test_denoise_wavelet</code>
<code>compute_threshold</code>	Compute the threshold, using one of the available algorithms	<code>test_threshold</code>
<code>compute_threshold_isodata</code>	Compute the threshold using the Isodata algorithm with default parameters	<code>test_threshold_isodata</code>
<code>compute_threshold_li</code>	Compute the threshold using the Li algorithm with default parameters	<code>test_threshold_li</code>
<code>compute_threshold_mean</code>	Compute the threshold using the Mean algorithm	<code>test_threshold_mean</code>
<code>compute_threshold_minimum</code>	Compute the threshold using the Minimum algorithm with default parameters	<code>test_threshold_minimum</code>
<code>compute_threshold_otsu</code>	Compute the threshold using the Otsu algorithm with default parameters	<code>test_threshold_otsu</code>

suite sur la page suivante

Tableau 3 – suite de la page précédente

Fonctions de calcul	Description	Fonction de test
<code>compute_threshold_triangle</code>	Compute the threshold using the Triangle algorithm with default parameters	<code>test_threshold_triangle</code>
<code>compute_threshold_yen</code>	Compute the threshold using the Yen algorithm with default parameters	<code>test_threshold_yen</code>

## 2.2 Fonctionnalités générales

Cette section décrit les fonctionnalités générales de DataLab, qui concernent à la fois les panneaux de traitement du signal et d'image.

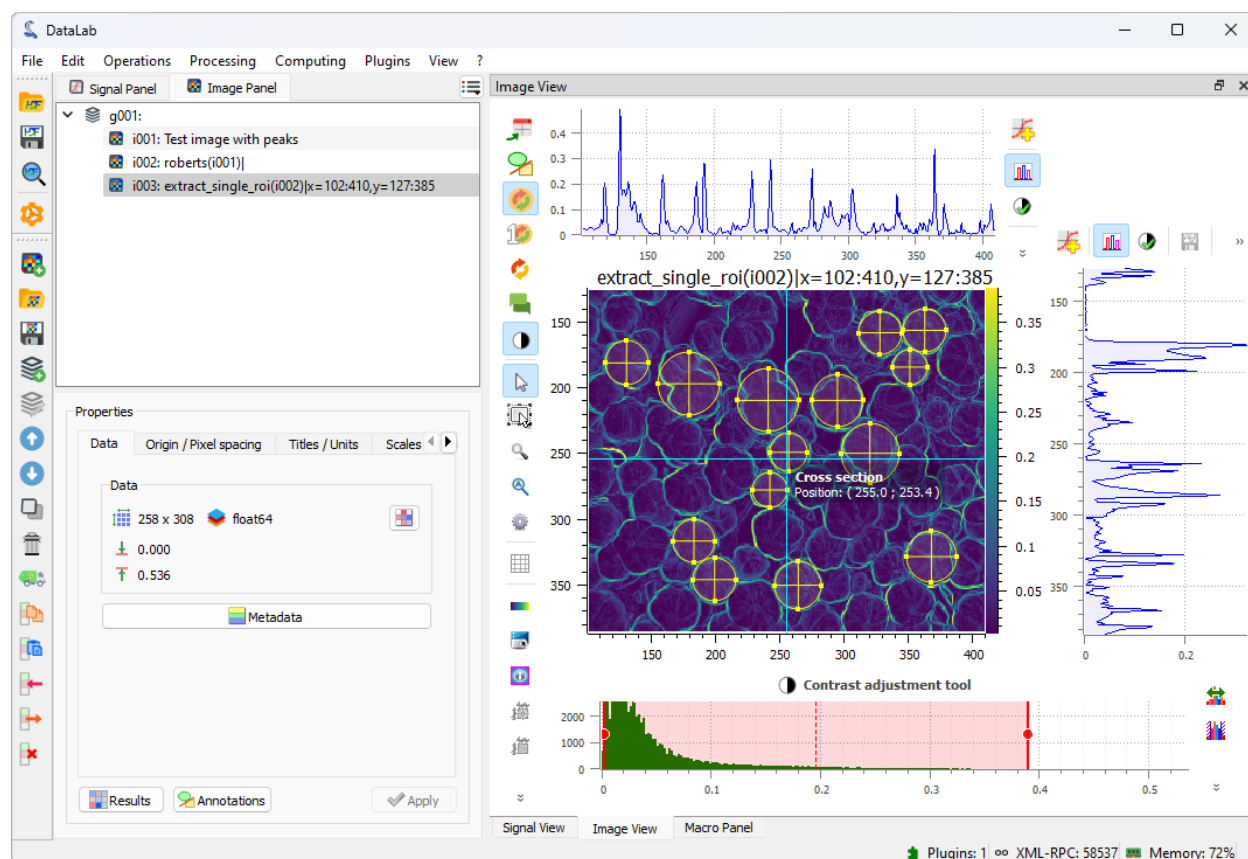


FIG. 1 – Fenêtre principale de DataLab

## 2.2.1 Espace de travail

### Concepts de base

Travailler avec DataLab est très simple. L'interface est intuitive et appréhendable facilement. La fenêtre principale est divisée en deux zones principales :

- La zone de gauche affiche la liste des jeux de données actuellement chargés dans DataLab, répartis sur deux onglets : **Signaux** et **Images**. L'utilisateur peut basculer entre les deux onglets en cliquant sur l'onglet correspondant : cela bascule la fenêtre principale vers le panneau correspondant, ainsi que le contenu du menu et de la barre d'outils. Sous la liste des jeux de données, une vue **Propriétés** affiche des informations sur le jeu de données actuellement sélectionné.
- La zone de droite affiche la visualisation du jeu de données actuellement sélectionné. La visualisation est mise à jour automatiquement lorsque l'utilisateur sélectionne un nouveau jeu de données dans la liste des jeux de données.

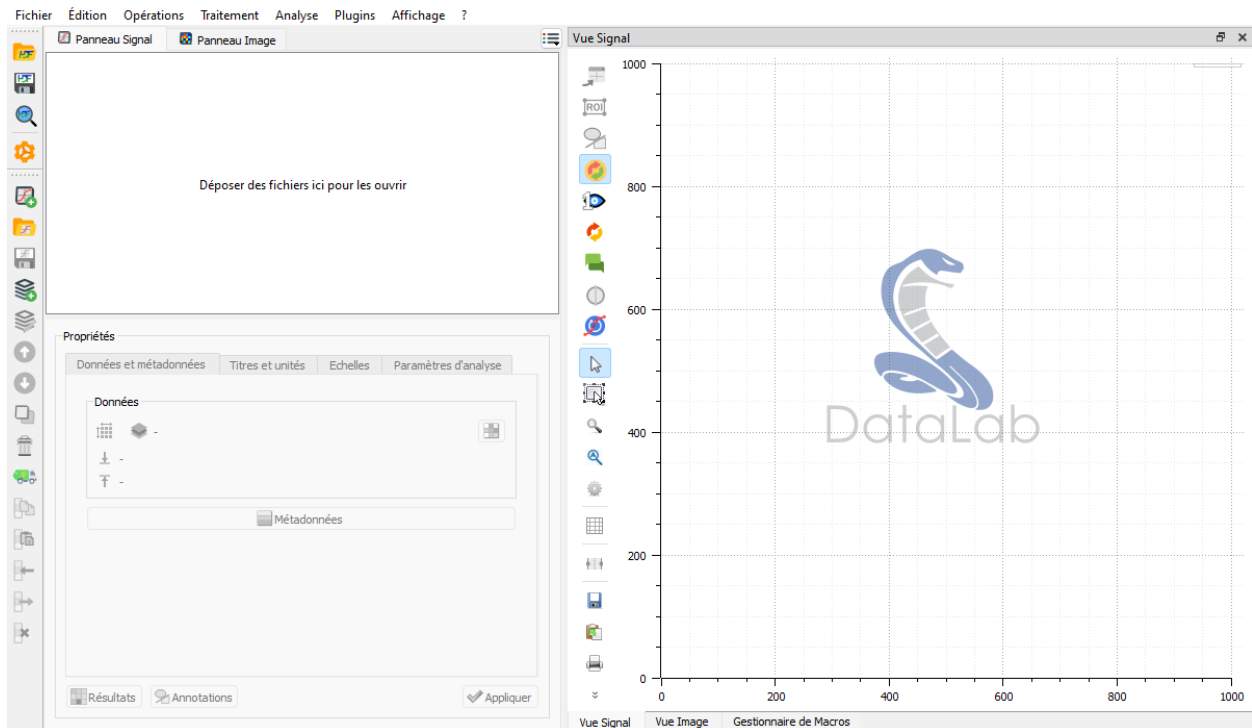


FIG. 2 – Fenêtre principale de DataLab, au démarrage.

### Modèle de données interne et espace de travail

DataLab a son propre modèle de données interne, dans lequel les jeux de données sont organisés autour d'une structure arborescente. Chaque panneau de la fenêtre principale correspond à une branche de l'arbre. Chaque jeu de données affiché dans les panneaux correspond à une feuille de l'arbre. À l'intérieur du jeu de données, les données sont organisées de manière orientée objet, avec un ensemble d'attributs et de méthodes. Le modèle de données est décrit plus en détail dans la section API (voir [cdl.obj](#)).

Pour chaque jeu de données (signal 1D ou image 2D), non seulement les données elles-mêmes sont stockées, mais aussi un ensemble de métadonnées, qui décrit les données ou la façon dont elles doivent être affichées. Les métadonnées sont stockées dans un dictionnaire, qui est accessible via l'attribut `metadata` du jeu de données (et peuvent également être parcourues dans la vue **Propriétés**, avec le bouton **Métadonnées**).

L'**Espace de travail** de DataLab est défini comme l'ensemble de tous les jeux de données actuellement chargés dans DataLab, dans les panneaux **Signaux** et **Images**.

## Chargement et enregistrement de l'espace de travail

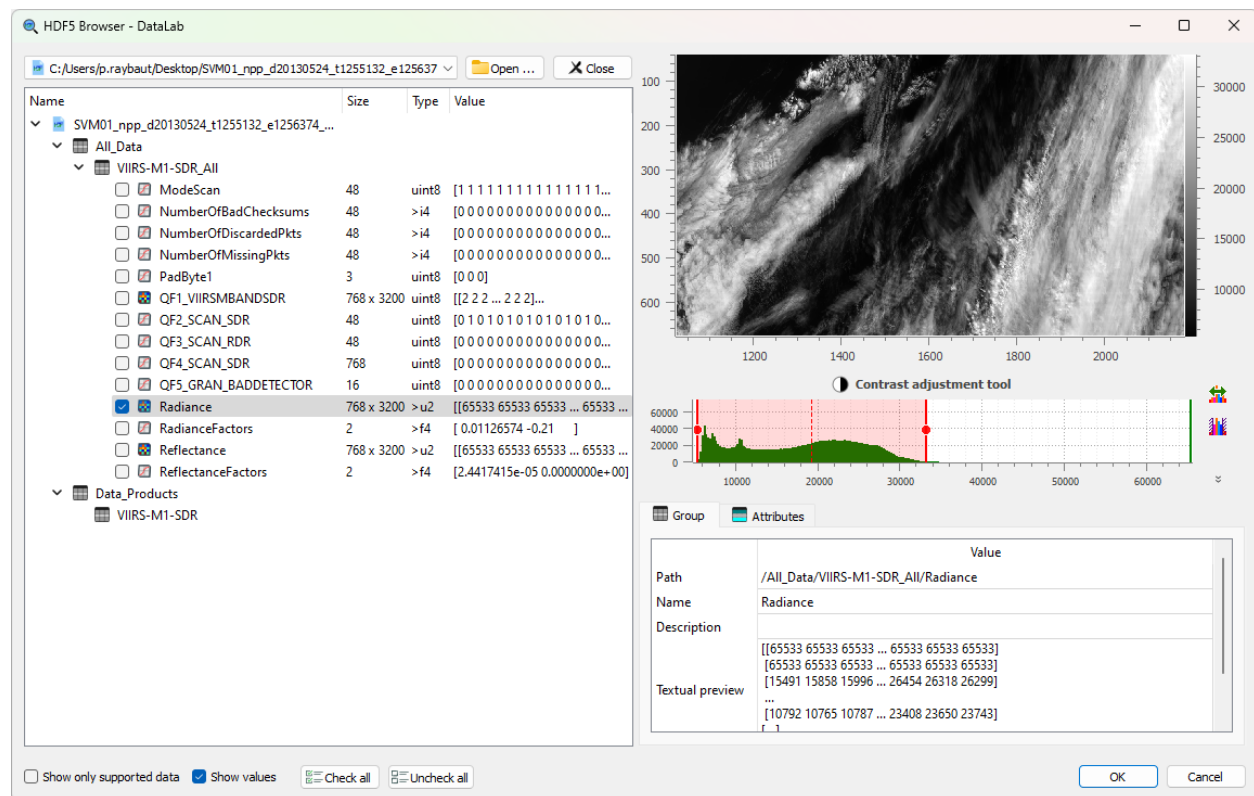
Les actions suivantes sont disponibles pour gérer l'espace de travail depuis le menu **Fichier** :

- **Ouvrir un fichier HDF5** : charger un espace de travail à partir d'un fichier HDF5.
- **Enregistrer dans un fichier HDF5** : enregistrer l'espace de travail actuel dans un fichier HDF5.
- **Parcourir un fichier HDF5** : ouvrir le *Explorateur HDF5* pour explorer le contenu d'un fichier HDF5 et importer des jeux de données dans l'espace de travail.

**Note** : Les jeux de données peuvent également être enregistrés ou chargés individuellement, en utilisant des formats de données tels que *.txt* ou *.npy* pour les signaux 1D (voir *Ouvrir un signal* pour la liste des formats pris en charge), ou *.tiff* ou *.dcm* pour les images 2D (voir *Ouvrir une image* pour la liste des formats pris en charge).

## 2.2.2 Explorateur HDF5

L'explorateur HDF5 est une boîte de dialogue modale permettant d'importer presque n'importe quelles données à 1 ou 2 dimensions dans l'espace de travail de DataLab. Dans certains cas, des métadonnées sont également importées.



Les données de type courbe ou image sont affichées sous la forme d'une vue hiérarchique dans le panneau de gauche, de même que les données scalaires (les valeurs scalaires sont simplement affichées à titre d'informations contextuelles et ne peuvent pas être importées dans l'espace de travail de DataLab).

Le panneau de droite affiche les données de courbe ou d'image sélectionnées. Il affiche également des informations sur le « Groupe » (chemin, description, etc.) et les « Attributs » (noms et valeurs).

**L'explorateur de fichiers HDF5 est très simple à utiliser :**

- Dans le panneau de gauche, sélectionner la courbe ou l'image à importer
- Les données sélectionnées peuvent être visualisées graphiquement dans le panneau de droite
- Cliquer sur « Cocher tout » pour importer toutes les données compatibles
- Valider ensuite en cliquant sur « OK »

**Note :** L'explorateur HDF5 peut être utilisé pour explorer plusieurs fichiers HDF5 à la fois, permettant ainsi d'importer des données de différents fichiers dans le même espace de travail de DataLab.

## 2.2.3 Macros

### Généralités

Plusieurs méthodes permettent d'étendre les fonctionnalités de DataLab (voir *Plugins* ou *Contrôle à distance*). La manière la plus simple de le faire est d'utiliser des macros. Les macros sont de petits scripts Python qui peuvent être exécutés depuis le « Gestionnaire de Macros » dans DataLab.

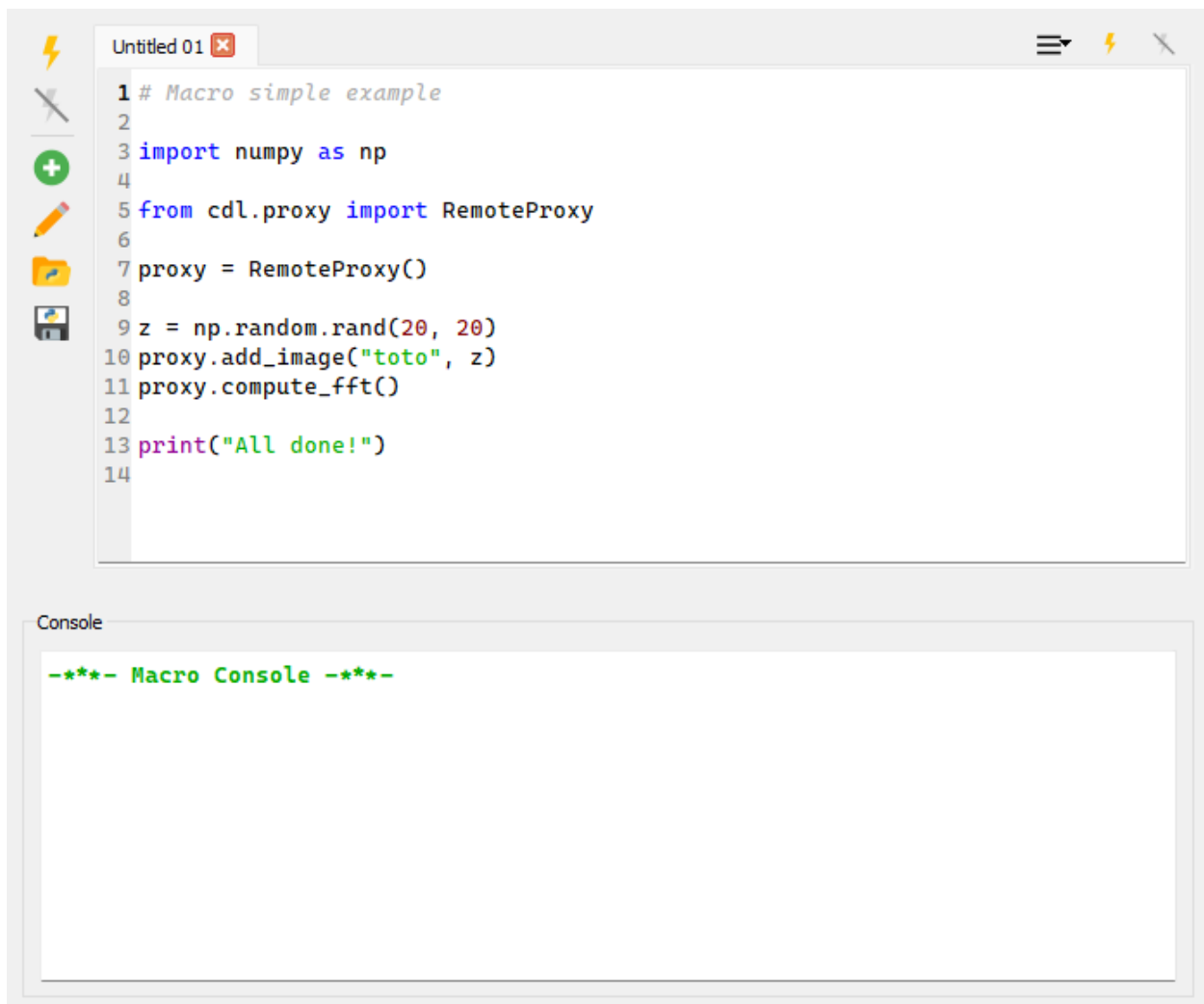


FIG. 3 – Le Gestionnaire de Macros dans DataLab.

Les macros peuvent être utilisées pour automatiser des tâches répétitives, ou pour créer de nouvelles fonctionnalités. Comme le système de plugins et de contrôle à distance, les macros reposent sur l'API de haut niveau de DataLab pour interagir avec l'application. Cela signifie que vous pouvez réutiliser les mêmes extraits de code dans les macros, les plugins et les scripts de contrôle à distance.

**Avertissement :** DataLab gère les macros comme des scripts Python. Cela signifie que vous pouvez utiliser toute la puissance de Python pour créer vos macros. Même si c'est une fonctionnalité puissante, cela signifie également que vous devez être prudent lorsque vous exécutez des macros à partir de sources inconnues, car elles peuvent potentiellement endommager votre système.

#### **Voir aussi :**

L'API de haut niveau de DataLab est documentée dans la section [API](#). Le système de plugins est documenté dans la section [Plugins](#), et le système de contrôle à distance est documenté dans la section [Contrôle à distance](#).

### **Fonctionnalités principales**

Le Gestionnaire de Macros est une interface simple pour :

- Créer de nouvelles macros, en utilisant le bouton « Nouvelle macro » .
- Renommer des macros existantes, en utilisant le bouton « Renommer macro » .
- Importer/exporter des macros depuis/vers des fichiers, en utilisant les boutons « Importer macro » et « Exporter macro » .
- Exécuter des macros, en utilisant le bouton « Exécuter macro » .
- Arrêter l'exécution d'une macro, en utilisant le bouton « Arrêter macro » .

Les macros sont intégrées dans l'espace de travail de DataLab, elles sont donc enregistrées avec le reste des données (c'est-à-dire avec les signaux et les images) lors de l'exportation de l'espace de travail vers un fichier HDF5. Cela signifie que vous pouvez partager vos macros avec d'autres utilisateurs simplement en partageant le fichier d'espace de travail.

---

**Note :** Les macros sont exécutées dans un processus séparé, elles ne bloqueront donc pas l'application principale de DataLab. Cela signifie que vous pouvez continuer à travailler avec DataLab pendant qu'une macro est en cours d'exécution et que *vous pouvez arrêter une macro à tout moment* en utilisant le bouton .

---

### **Exemple**

Pour un exemple détaillé de création d'une macro, voir le tutoriel [Prototypage d'une chaîne de traitement personnalisée](#).

### **2.2.4 Contrôle à distance**

DataLab peut être contrôlé à distance en utilisant le protocole [XML-RPC](#) qui est nativement supporté par Python (et beaucoup d'autres langages). Le contrôle à distance permet d'accéder aux principales fonctionnalités de DataLab à partir d'un processus séparé.

---

**Note :** Si vous recherchez une solution alternative légère pour contrôler DataLab à distance (c'est-à-dire sans avoir à installer l'ensemble du package DataLab et ses dépendances sur votre environnement), veuillez consulter le package [DataLab Simple Client](#) (`pip install cdlclient`).

---

## Depuis un IDE

DataLab peut être contrôlé à distance depuis un IDE (par exemple [Spyder](#) ou tout autre IDE, ou même un Jupyter Notebook) qui exécute un script Python. Il permet de se connecter à une instance de DataLab en cours d'exécution, d'ajouter un signal et une image, puis d'exécuter des calculs. Cette fonctionnalité est exposée par la classe `RemoteProxy` fournie dans le module `cdl.proxy`.

## Depuis une application tierce

DataLab peut également être contrôlé à distance depuis une application tierce, dans le même but.

Si l'application tierce est écrite en Python 3, elle peut directement utiliser la classe `RemoteProxy` comme mentionné ci-dessus. Depuis un autre langage, c'est également réalisable, mais cela nécessite d'implémenter un client XML-RPC dans ce langage en utilisant les mêmes méthodes de serveur proxy que dans la classe `RemoteProxy`.

Les données (signaux et images) peuvent également être échangées entre DataLab et l'application cliente distante, dans les deux sens.

L'application cliente distante peut être écrite dans n'importe quel langage qui prend en charge XML-RPC. Par exemple, il est possible d'écrire une application cliente distante en Python, Java, C++, C#, etc. L'application cliente distante peut être une application graphique ou une application en ligne de commande.

L'application cliente distante peut être exécutée sur le même ordinateur que DataLab ou sur un ordinateur différent. Dans ce dernier cas, l'application cliente distante doit connaître l'adresse IP de l'ordinateur exécutant DataLab.

L'application cliente distante peut être exécutée avant ou après DataLab. Dans ce dernier cas, l'application cliente distante doit essayer de se connecter à DataLab jusqu'à ce qu'elle réussisse.

## Fonctionnalités prises en charge

Les fonctionnalités prises en charge sont les suivantes :

- Basculer vers le panneau de signal ou d'image
- Supprimer tous les signaux et images
- Enregistrer la session en cours dans un fichier HDF5
- Ouvrir des fichiers HDF5 dans la session en cours
- Parcourir un fichier HDF5
- Ouvrir un signal ou une image à partir d'un fichier
- Ajouter un signal
- Ajouter une image
- Obtenir la liste des objets
- Exécuter un calcul avec des paramètres

---

**Note :** Les objets signal et image sont décrits dans cette section : *Modèle de données interne*.

---

Quelques exemples sont fournis pour aider à implémenter une telle communication entre votre application et DataLab :

- Voir le module : `cdl.tests.remoteclient_app`
- Voir le module : `cdl.tests.remoteclient_unit`

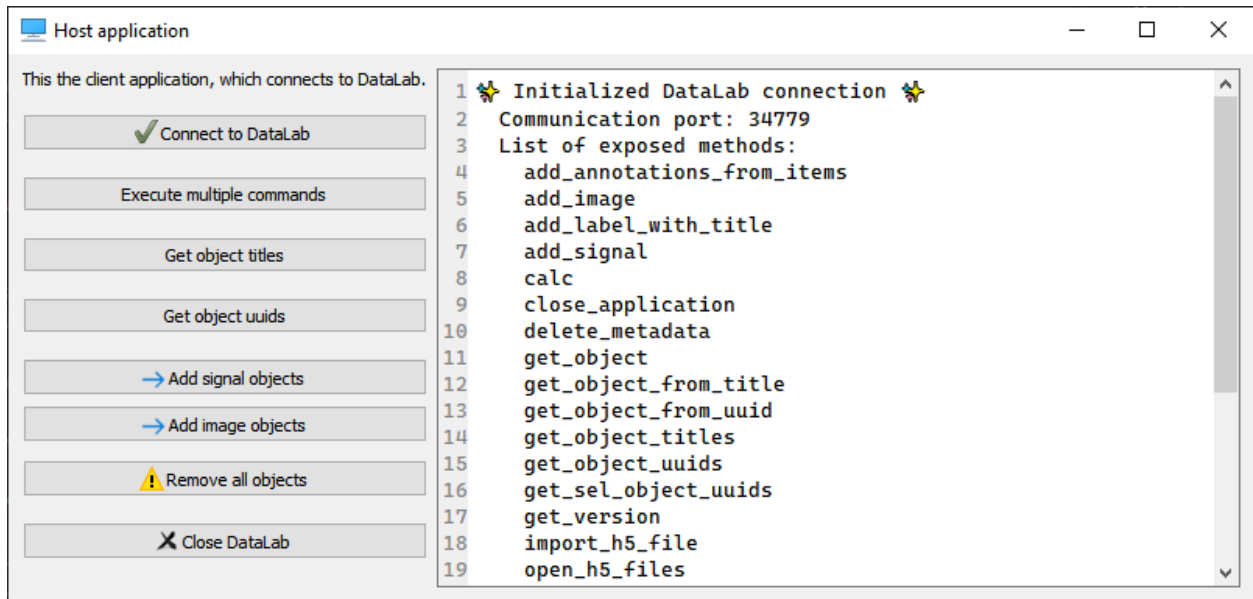


FIG. 4 – Capture d'écran du test de l'application cliente distante (`cdl.tests.remoteclient_app`)

## Exemples

Lorsque vous utilisez Python 3, vous pouvez utiliser directement la classe *RemoteProxy* comme dans les exemples cités ci-dessus ou ci-dessous.

Voici un exemple en Python 3 d'un script qui se connecte à une instance de DataLab en cours d'exécution, ajoute un signal et une image, puis exécute des calculs (la structure de cellule du script le rend pratique à utiliser dans l'IDE Spyder) :

```
"""
Example of remote control of DataLab current session,
from a Python script running outside DataLab (e.g. in Spyder)

Created on Fri May 12 12:28:56 2023

@author: p.raybaut
"""

# %% Importing necessary modules

# NumPy for numerical array computations:
import numpy as np

# DataLab remote control client:
from cdlclient import SimpleRemoteProxy as RemoteProxy

# %% Connecting to DataLab current session

proxy = RemoteProxy()

# %% Executing commands in DataLab (...)
```

(suite sur la page suivante)

(suite de la page précédente)

```

z = np.random.rand(20, 20)
proxy.add_image("toto", z)

# %% Executing commands in DataLab (...)

proxy.toggle_auto_refresh(False) # Turning off auto-refresh
x = np.array([1.0, 2.0, 3.0])
y = np.array([4.0, 5.0, -1.0])
proxy.add_signal("toto", x, y)

# %% Executing commands in DataLab (...)

proxy.compute_derivative()
proxy.toggle_auto_refresh(True) # Turning on auto-refresh

# %% Executing commands in DataLab (...)

proxy.set_current_panel("image")

# %% Executing a lot of commands without refreshing DataLab

z = np.random.rand(400, 400)
proxy.add_image("foobar", z)
with proxy.context_no_refresh():
    for _idx in range(100):
        proxy.compute_fft()

```

Voici une réimplémentation de cette classe en Python 2.7 :

```

# Copyright (c) DataLab Platform Developers, BSD 3-Clause license, see LICENSE file.

"""
DataLab remote controlling class for Python 2.7
"""

import io
import os
import os.path as osp
import socket
import sys

import ConfigParser as cp
import numpy as np
from guidata.userconfig import get_config_dir
from xmlrpclib import Binary, ServerProxy

def array_to_rpcbinary(data):
    """Convert NumPy array to XML-RPC Binary object, with shape and dtype"""
    dbytes = io.BytesIO()
    np.save(dbytes, data, allow_pickle=False)

```

(suite sur la page suivante)

```

    return Binary(dbytes.getvalue())

def get_cdl_xmlrpc_port():
    """Return DataLab current XML-RPC port"""
    if sys.platform == "win32" and "HOME" in os.environ:
        os.environ.pop("HOME") # Avoid getting old WinPython settings dir
    fname = osp.join(get_config_dir(), ".DataLab", "DataLab.ini")
    ini = cp.ConfigParser()
    ini.read(fname)
    try:
        return ini.get("main", "rpc_server_port")
    except (cp.NoSectionError, cp.NoOptionError):
        raise ConnectionRefusedError("DataLab has not yet been executed")

class RemoteClient(object):
    """Object representing a proxy/client to DataLab XML-RPC server"""

    def __init__(self):
        self.port = None
        self.serverproxy = None

    def connect(self, port=None):
        """Connect to DataLab XML-RPC server"""
        if port is None:
            port = get_cdl_xmlrpc_port()
        self.port = port
        url = "http://127.0.0.1:" + port
        self.serverproxy = ServerProxy(url, allow_none=True)
        try:
            self.get_version()
        except socket.error:
            raise ConnectionRefusedError("DataLab is currently not running")

    def get_version(self):
        """Return DataLab public version"""
        return self.serverproxy.get_version()

    def close_application(self):
        """Close DataLab application"""
        self.serverproxy.close_application()

    def raise_window(self):
        """Raise DataLab window"""
        self.serverproxy.raise_window()

    def get_current_panel(self):
        """Return current panel"""
        return self.serverproxy.get_current_panel()

    def set_current_panel(self, panel):

```

(suite de la page précédente)

```

        """Switch to panel"""
        self.serverproxy.set_current_panel(panel)

    def reset_all(self):
        """Reset all application data"""
        self.serverproxy.reset_all()

    def toggle_auto_refresh(self, state):
        """Toggle auto refresh state"""
        self.serverproxy.toggle_auto_refresh(state)

    def toggle_show_titles(self, state):
        """Toggle show titles state"""
        self.serverproxy.toggle_show_titles(state)

    def save_to_h5_file(self, filename):
        """Save to a DataLab HDF5 file"""
        self.serverproxy.save_to_h5_file(filename)

    def open_h5_files(self, h5files, import_all, reset_all):
        """Open a DataLab HDF5 file or import from any other HDF5 file"""
        self.serverproxy.open_h5_files(h5files, import_all, reset_all)

    def import_h5_file(self, filename, reset_all):
        """Open DataLab HDF5 browser to Import HDF5 file"""
        self.serverproxy.import_h5_file(filename, reset_all)

    def load_from_files(self, filenames):
        """Open objects from files in current panel (signals/images)"""
        self.serverproxy.load_from_files(filenames)

    def add_signal(
        self, title, xdata, ydata, xunit=None, yunit=None, xlabel=None, ylabel=None
    ):
        """Add signal data to DataLab"""
        xbinary = array_to_rpcbinary(xdata)
        ybinary = array_to_rpcbinary(ydata)
        p = self.serverproxy
        return p.add_signal(title, xbinary, ybinary, xunit, yunit, xlabel, ylabel)

    def add_image(
        self,
        title,
        data,
        xunit=None,
        yunit=None,
        zunit=None,
        xlabel=None,
        ylabel=None,
        zlabel=None,
    ):
        """Add image data to DataLab"""

```

(suite sur la page suivante)

(suite de la page précédente)

```

        zbinary = array_to_rpcbinary(data)
        p = self.serverproxy
        return p.add_image(title, zbinary, xunit, yunit, zunit, xlabel, ylabel, zlabel)

    def get_object_titles(self, panel=None):
        """Get object (signal/image) list for current panel"""
        return self.serverproxy.get_object_titles(panel)

    def get_object(self, nb_id_title=None, panel=None):
        """Get object (signal/image) by number, id or title"""
        return self.serverproxy.get_object(nb_id_title, panel)

    def get_object_uuids(self, panel=None):
        """Get object (signal/image) list for current panel"""
        return self.serverproxy.get_object_uuids(panel)

def test_remote_client():
    """DataLab Remote Client test"""
    cdl = RemoteClient()
    cdl.connect()
    data = np.array([[3, 4, 5], [7, 8, 0]], dtype=np.uint16)
    cdl.add_image("toto", data)

if __name__ == "__main__":
    test_remote_client()

```

### Boîte de dialogue de connexion

Le package DataLab fournit également une boîte de dialogue de connexion qui peut être utilisée pour se connecter à une instance de DataLab en cours d'exécution. Elle est exposée par la classe `cdl.widgets.connection.ConnectionDialog`.

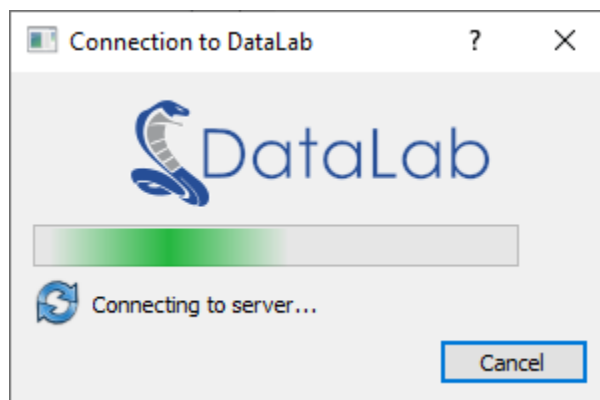


FIG. 5 – Capture d'écran de la boîte de dialogue de connexion (`cdl.widgets.connection.ConnectionDialog`)

Exemple d'utilisation :

```
# Copyright (c) DataLab Platform Developers, BSD 3-Clause license, see LICENSE file.

"""
DataLab Remote client connection dialog example
"""

# guitest: show,skip

from guidata.qthelpers import qt_app_context
from qtpy import QtWidgets as QW

from cdl.proxy import RemoteProxy
from cdl.widgets.connection import ConnectionDialog

def test_dialog():
    """Test connection dialog"""
    proxy = RemoteProxy(autoconnect=False)
    with qt_app_context():
        dlg = ConnectionDialog(proxy.connect)
        if dlg.exec():
            QW.QMessageBox.information(None, "Connection", "Successfully connected")
        else:
            QW.QMessageBox.critical(None, "Connection", "Connection failed")

if __name__ == "__main__":
    test_dialog()
```

## API publique : client distant

### class cdl.core.remote.RemoteClient

Objet représentant un proxy/client vers le serveur XML-RPC de DataLab. Cet objet est utilisé pour appeler les fonctions de DataLab à partir d'un script Python.

### Exemples

Voici un exemple simple de l'utilisation de RemoteClient dans un script Python ou dans un notebook Jupyter :

```
>>> from cdl.core.remote import RemoteClient
>>> proxy = RemoteClient()
>>> proxy.connect()
Connecting to DataLab XML-RPC server...OK (port: 28867)
>>> proxy.get_version()
'1.0.0'
>>> proxy.add_signal("toto", np.array([1., 2., 3.]), np.array([4., 5., -1.]))
True
>>> proxy.get_object_titles()
['toto']
>>> proxy["toto"]
<cdl.core.model.signal.SignalObj at 0x7f7f1c0b4a90>
```

(suite sur la page suivante)

(suite de la page précédente)

```
>>> proxy[1]
<cdl.core.model.signal.SignalObj at 0x7f7f1c0b4a90>
>>> proxy[1].data
array([1., 2., 3.]
```

**connect**(port : *str* | *None* = *None*, timeout : *float* | *None* = *None*, retries : *int* | *None* = *None*) → *None*

Tentative de connexion au serveur XML-RPC de DataLab.

#### Paramètres

- **port** – Port XML-RPC auquel se connecter. Si non spécifié, le port est automatiquement récupéré à partir de la configuration de DataLab.
- **timeout** – Délai d’attente en secondes. Par défaut, 5.0.
- **retries** – Nombre de tentatives. Par défaut, 10.

#### Lève

- **ConnectionRefusedError** – Impossible de se connecter à DataLab
- **ValueError** – Délai d’attente non valide (doit être >= 0.0)
- **ValueError** – Nombre de tentatives non valide (doit être >= 1)

**disconnect**() → *None*

Déconnexion du serveur XML-RPC de DataLab.

**is\_connected**() → *bool*

Renvoie True si connecté au serveur XML-RPC de DataLab.

**get\_method\_list**() → *list[str]*

Renvoie la liste des méthodes disponibles.

**add\_signal**(title : *str*, xdata : *ndarray*, ydata : *ndarray*, xunit : *str* | *None* = *None*, yunit : *str* | *None* = *None*, xlabel : *str* | *None* = *None*, ylabel : *str* | *None* = *None*) → *bool*

Ajouter des données de signal à DataLab.

#### Paramètres

- **title** – Titre du signal
- **xdata** – Données X
- **ydata** – Données Y
- **xunit** – Unité X. Par défaut, *None*.
- **yunit** – Unité Y. Par défaut, *None*.
- **xlabel** – Libellé X. Par défaut, *None*.
- **ylabel** – Libellé Y. Par défaut, *None*.

#### Renvoie

True si le signal a été ajouté avec succès, False sinon

#### Lève

- **ValueError** – Type de données xdata non valide
- **ValueError** – Type de données ydata non valide

**add\_image**(title : *str*, data : *ndarray*, xunit : *str* | *None* = *None*, yunit : *str* | *None* = *None*, zunit : *str* | *None* = *None*, xlabel : *str* | *None* = *None*, ylabel : *str* | *None* = *None*, zlabel : *str* | *None* = *None*) → *bool*

Ajouter des données d’image à DataLab.

#### Paramètres

- **title** – Titre de l’image
- **data** – Données de l’image
- **xunit** – Unité X. Par défaut, *None*.
- **yunit** – Unité Y. Par défaut, *None*.
- **zunit** – Unité Z. Par défaut, *None*.
- **xlabel** – Libellé X. Par défaut, *None*.

- **ylabel** – Libellé Y. Par défaut, None.
- **zlabel** – Libellé Z. Par défaut, None.

**Renvoie**

True si l'image a été ajoutée avec succès, False sinon

**Lève**

**ValueError** – Type de données non valide

**add\_object**(*obj* : *SignalObj* | *ImageObj*) → *None*

Ajouter un objet à DataLab.

**Paramètres**

**obj** – Objet signal ou image

**calc**(*name* : *str*, *param* : *DataSet* | *None* = *None*) → *None*

Appeler la fonction de calcul *name* dans le processeur du panneau en cours.

**Paramètres**

- **name** – Nom de la fonction de calcul
- **param** – Paramètre de la fonction de calcul. Par défaut, None.

**Lève**

**ValueError** – unknown function

**get\_object**(*nb\_id\_title* : *int* | *str* | *None* = *None*, *panel* : *str* | *None* = *None*) → *SignalObj* | *ImageObj*

Obtenir un objet (signal/image) à partir de l'index.

**Paramètres**

- **nb\_id\_title** – Numéro d'objet, ou identifiant d'objet, ou titre d'objet. Par défaut, None (objet en cours).
- **panel** – Nom du panneau. Par défaut, None (panneau en cours).

**Renvoie**

Objet

**Lève**

**KeyError** – si l'objet n'est pas trouvé

**get\_object\_shapes**(*nb\_id\_title* : *int* | *str* | *None* = *None*, *panel* : *str* | *None* = *None*) → *list*

Obtenir les formes géométriques des éléments de tracé associés à l'objet (signal/image).

**Paramètres**

- **nb\_id\_title** – Numéro d'objet, ou identifiant d'objet, ou titre d'objet. Par défaut, None (objet en cours).
- **panel** – Nom du panneau. Par défaut, None (panneau en cours).

**Renvoie**

Liste des formes géométriques des éléments de tracé

**add\_annotations\_from\_items**(*items* : *list*, *refresh\_plot* : *bool* = *True*, *panel* : *str* | *None* = *None*) → *None*

Ajouter des annotations d'objet (éléments de tracé d'annotation).

**Paramètres**

- **items** – éléments de tracé d'annotation
- **refresh\_plot** – rafraîchir le tracé. Par défaut, True.
- **panel** – nom du panneau (valeurs valides : « signal », « image »). Si None, le panneau en cours est utilisé.

**add\_label\_with\_title**(*title* : *str* | *None* = *None*, *panel* : *str* | *None* = *None*) → *None*

Ajouter une étiquette avec le titre de l'objet sur le tracé associé

**Paramètres**

- **title** – Titre de l'étiquette. Par défaut, None. Si None, le titre est le titre de l'objet.
- **panel** – nom du panneau (valeurs valides : « signal », « image »). Si None, le panneau en cours est utilisé.

**close\_application()** → *None*

Fermer l'application DataLab

**context\_no\_refresh()** → *Generator[None, None, None]*

Renvoie un gestionnaire de contexte pour désactiver temporairement le rafraîchissement automatique.

**Renvoie**

Gestionnaire de contexte

### Exemple

```
>>> with proxy.context_no_refresh():
...     proxy.add_image("image1", data1)
...     proxy.compute_fft()
...     proxy.compute_wiener()
...     proxy.compute_ifft()
...     # Auto refresh is disabled during the above operations
```

**delete\_metadata(refresh\_plot : bool = True, keep\_roi : bool = False)** → *None*

Supprimer les métadonnées des objets sélectionnés

**Paramètres**

— **refresh\_plot** – Rafraîchir le tracé. Par défaut, True.

— **keep\_roi** – Conserver la ROI. Par défaut, False.

**get\_current\_panel()** → *str*

Renvoie le nom du panneau en cours.

**Renvoie**

« signal », « image », « macro »))

**Type renvoyé**

Panel name (valid values

**get\_group\_titles\_with\_object\_infos()** → *tuple[list[str], list[list[str]], list[list[str]]]*

Renvoie les titres des groupes et les listes des uuids et titres des objets internes.

**Renvoie**

titres des groupes, listes des uuids et titres des objets internes

**Type renvoyé**

Tuple

**get\_object\_titles(panel : str | None = None)** → *list[str]*

Obtenir la liste des objets (signal/image) pour le panneau en cours. Les objets sont triés par numéro de groupe et index d'objet dans le groupe.

**Paramètres**

**panel** – nom du panneau (valeurs valides : « signal », « image », « macro »). Si None, le panneau de données en cours est utilisé (c'est-à-dire le panneau de signal ou d'image).

**Renvoie**

Liste des titres des objets

**Lève**

**ValueError** – si le panneau n'est pas trouvé

**get\_object\_uuids(panel : str | None = None)** → *list[str]*

Obtenir la liste des uuids des objets (signal/image) pour le panneau en cours. Les objets sont triés par numéro de groupe et index d'objet dans le groupe.

**Paramètres**

**panel** – nom du panneau (valeurs valides : « signal », « image »). Si None, le panneau en cours est utilisé.

**Renvoie**

List of object uuids

**Lève**

**ValueError** – si le panneau n’est pas trouvé

**classmethod** **get\_public\_methods()** → list[str]

Renvoie toutes les méthodes publiques de la classe, sauf elle-même.

**Renvoie**

Liste des méthodes publiques

**get\_sel\_object\_uuids**(include\_groups : bool = False) → list[str]

Renvoie les uuids des objets sélectionnés.

**Paramètres**

**include\_groups** – Si True, renvoie également les objets des groupes sélectionnés.

**Renvoie**

Liste des uuids des objets sélectionnés.

**get\_version()** → str

Return DataLab public version.

**Renvoie**

Version de DataLab

**import\_h5\_file**(filename : str, reset\_all : bool | None = None) → None

Ouvrir le navigateur HDF5 de DataLab pour importer un fichier HDF5.

**Paramètres**

— **filename** – Nom du fichier HDF5

— **reset\_all** – Réinitialiser toutes les données de l’application. Par défaut, None.

**import\_macro\_from\_file**(filename : str) → None

Importer une macro à partir d’un fichier

**Paramètres**

**filename** – Nom du fichier.

**load\_from\_files**(filenames : list[str]) → None

Ouvrir des objets à partir de fichiers dans le panneau en cours

**Paramètres**

**filenames** – liste des noms de fichiers

**open\_h5\_files**(h5files : list[str] | None = None, import\_all : bool | None = None, reset\_all : bool | None = None) → None

Ouvrir un fichier HDF5 de DataLab ou importer à partir de tout autre fichier HDF5.

**Paramètres**

— **h5files** – Liste des fichiers HDF5 à ouvrir. Par défaut, None.

— **import\_all** – Importer tous les objets à partir des fichiers HDF5. Par défaut, None.

— **reset\_all** – Réinitialiser toutes les données de l’application. Par défaut, None.

**raise\_window()** → None

Faire apparaître la fenêtre de DataLab

**reset\_all()** → None

Réinitialiser toutes les données de l’application

**run\_macro**(*number\_or\_title* : *int* | *str* | *None* = *None*) → *None*

Exécute la macro.

**Paramètres**

**number\_or\_title** – Numéro de macro, ou titre de macro. Par défaut, *None* (macro en cours).

**Lève**

**ValueError** – si la macro n’est pas trouvée

**save\_to\_h5\_file**(*filename* : *str*) → *None*

Enregistrer dans un fichier HDF5 de DataLab.

**Paramètres**

**filename** – Nom du fichier HDF5

**select\_groups**(*selection* : *list*[*int* | *str*] | *None* = *None*, *panel* : *str* | *None* = *None*) → *None*

Sélectionner des groupes dans le panneau en cours.

**Paramètres**

— **selection** – Liste des numéros de groupe (1 à N), ou liste des uuids de groupe, ou *None* pour sélectionner tous les groupes. Par défaut, *None*.

— **panel** – nom du panneau (valeurs valides : « signal », « image »). Si *None*, le panneau en cours est utilisé. Par défaut, *None*.

**select\_objects**(*selection* : *list*[*int* | *str*], *panel* : *str* | *None* = *None*) → *None*

Sélectionner des objets dans le panneau en cours.

**Paramètres**

— **selection** – Liste des numéros d’objet (1 à N) ou uuids à sélectionner

— **panel** – nom du panneau (valeurs valides : « signal », « image »). Si *None*, le panneau en cours est utilisé. Par défaut, *None*.

**set\_current\_panel**(*panel* : *str*) → *None*

Basculer vers le panneau.

**Paramètres**

**panel** – Nom du panneau (valeurs valides : « signal », « image », « macro »)

**stop\_macro**(*number\_or\_title* : *int* | *str* | *None* = *None*) → *None*

Arrête la macro.

**Paramètres**

**number\_or\_title** – Numéro de macro, ou titre de macro. Par défaut, *None* (macro en cours).

**Lève**

**ValueError** – si la macro n’est pas trouvée

**toggle\_auto\_refresh**(*state* : *bool*) → *None*

Basculer l’état du rafraîchissement automatique.

**Paramètres**

**state** – État du rafraîchissement automatique

**toggle\_show\_titles**(*state* : *bool*) → *None*

Basculer l’état d’affichage des titres.

**Paramètres**

**state** – État d’affichage des titres

## API publique : méthodes supplémentaires

Les méthodes de la classe de contrôle à distance (en utilisant le proxy ou le client distant) peuvent être complétées par des méthodes supplémentaires qui sont ajoutées dynamiquement à l'exécution. Ce mécanisme permet d'accéder aux méthodes des processeurs de DataLab (voir `cdl.core.gui.processor`).

### Processeur de signal

Lorsque vous travaillez avec des signaux, les méthodes de `cdl.core.gui.processor.signal.SignalProcessor` peuvent être accédées.

**class** `cdl.core.gui.processor.signal.SignalProcessor`(*panel* : `SignalPanel` | `ImagePanel`, *plotwidget* : `PlotWidget`)

Objet de traitement de signal : opérations, traitement, analyse

**compute\_sum**() → `None`

Compute sum with `cdl.computation.signal.compute_addition()`

**compute\_addition\_constant**(*param* : `ConstantParam` | `None = None`) → `None`

Compute sum with a constant with `cdl.computation.signal.compute_addition_constant()`

**compute\_average**() → `None`

Compute average with `cdl.computation.signal.compute_addition()` and divide by the number of signals

**compute\_product**() → `None`

Compute product with `cdl.computation.signal.compute_product()`

**compute\_product\_constant**(*param* : `ConstantParam` | `None = None`) → `None`

Compute product with a constant with `cdl.computation.signal.compute_product_constant()`

**compute\_swap\_axes**() → `None`

Swap data axes with `cdl.computation.signal.compute_swap_axes()`

**compute\_abs**() → `None`

Compute absolute value with `cdl.computation.signal.compute_abs()`

**compute\_re**() → `None`

Compute real part with `cdl.computation.signal.compute_re()`

**compute\_im**() → `None`

Compute imaginary part with `cdl.computation.signal.compute_im()`

**compute\_astype**(*param* : `DataTypeSParam` | `None = None`) → `None`

Convert data type with `cdl.computation.signal.compute_astype()`

**compute\_log10**() → `None`

Compute Log10 with `cdl.computation.signal.compute_log10()`

**compute\_exp**() → `None`

Compute Log10 with `cdl.computation.signal.compute_exp()`

**compute\_sqrt**() → `None`

Compute square root with `cdl.computation.signal.compute_sqrt()`

**compute\_power**(*param* : `PowerParam` | `None = None`) → `None`

Compute power with `cdl.computation.signal.compute_power()`

**compute\_arithmetic**(obj2 : SignalObj | None = None, param : ArithmeticParam | None = None) → None  
Compute arithmetic operation between two signals with `cdl.computation.signal.compute_arithmetic()`

**compute\_difference**(obj2 : SignalObj | list[SignalObj] | None = None) → None  
Compute difference between two signals with `cdl.computation.signal.compute_difference()`

**compute\_difference\_constant**(param : ConstantParam | None = None) → None  
Compute difference with a constant with `cdl.computation.signal.compute_difference_constant()`

**compute\_quadratic\_difference**(obj2 : SignalObj | list[SignalObj] | None = None) → None  
Compute quadratic difference between two signals with `cdl.computation.signal.compute_quadratic_difference()`

**compute\_division**(obj2 : SignalObj | list[SignalObj] | None = None) → None  
Compute division between two signals with `cdl.computation.signal.compute_division()`

**compute\_division\_constant**(param : ConstantParam | None = None) → None  
Compute division by a constant with `cdl.computation.signal.compute_division_constant()`

**compute\_peak\_detection**(param : PeakDetectionParam | None = None) → None  
Detect peaks from data with `cdl.computation.signal.compute_peak_detection()`

**compute\_reverse\_x**() → None  
Reverse X axis with `cdl.computation.signal.compute_reverse_x()`

**compute\_normalize**(param : NormalizeParam | None = None) → None  
Normalize data with `cdl.computation.signal.compute_normalize()`

**compute\_derivative**() → None  
Compute derivative with `cdl.computation.signal.compute_derivative()`

**compute\_integral**() → None  
Compute integral with `cdl.computation.signal.compute_integral()`

**compute\_calibration**(param : XYCalibrateParam | None = None) → None  
Compute data linear calibration with `cdl.computation.signal.compute_calibration()`

**compute\_clip**(param : ClipParam | None = None) → None  
Compute maximum data clipping with `cdl.computation.signal.compute_clip()`

**compute\_offset\_correction**(param : ROIIDParam | None = None) → None  
Compute offset correction with `cdl.computation.signal.compute_offset_correction()`

**compute\_gaussian\_filter**(param : GaussianParam | None = None) → None  
Compute gaussian filter with `cdl.computation.signal.compute_gaussian_filter()`

**compute\_moving\_average**(param : MovingAverageParam | None = None) → None  
Compute moving average with `cdl.computation.signal.compute_moving_average()`

**compute\_moving\_median**(param : MovingMedianParam | None = None) → None  
Compute moving median with `cdl.computation.signal.compute_moving_median()`

**compute\_wiener**() → None  
Compute Wiener filter with `cdl.computation.signal.compute_wiener()`

**compute\_lowpass**(param : LowPassFilterParam | None = None) → None  
Compute high-pass filter with `cdl.computation.signal.compute_filter()`

**compute\_highpass**(*param* : HighPassFilterParam | None = None) → None  
 Compute high-pass filter with `cdl.computation.signal.compute_filter()`

**compute\_bandpass**(*param* : BandPassFilterParam | None = None) → None  
 Compute band-pass filter with `cdl.computation.signal.compute_filter()`

**compute\_bandstop**(*param* : BandStopFilterParam | None = None) → None  
 Compute band-stop filter with `cdl.computation.signal.compute_filter()`

**compute\_fft**(*param* : FFTParam | None = None) → None  
 Compute FFT with `cdl.computation.signal.compute_fft()`

**compute\_ifft**(*param* : FFTParam | None = None) → None  
 Compute iFFT with `cdl.computation.signal.compute_ifft()`

**compute\_magnitude\_spectrum**(*param* : SpectrumParam | None = None) → None  
 Compute magnitude spectrum with `cdl.computation.signal.compute_magnitude_spectrum()`

**compute\_phase\_spectrum**() → None  
 Compute phase spectrum with `cdl.computation.signal.compute_phase_spectrum()`

**compute\_psd**(*param* : SpectrumParam | None = None) → None  
 Compute power spectral density with `cdl.computation.signal.compute_psd()`

**compute\_interpolation**(*obj2* : SignalObj | None = None, *param* : InterpolationParam | None = None)  
 Compute interpolation with `cdl.computation.signal.compute_interpolation()`

**compute\_resampling**(*param* : ResamplingParam | None = None)  
 Compute resampling with `cdl.computation.signal.compute_resampling()`

**compute\_detrending**(*param* : DetrendingParam | None = None)  
 Compute detrending with `cdl.computation.signal.compute_detrending()`

**compute\_convolution**(*obj2* : SignalObj | None = None) → None  
 Compute convolution with `cdl.computation.signal.compute_convolution()`

**compute\_windowing**(*param* : WindowingParam | None = None) → None  
 Compute windowing with `cdl.computation.signal.compute_windowing()`

**compute\_polyfit**(*param* : PolynomialFitParam | None = None) → None  
 Calculer la courbe de régression polynomiale

**compute\_fit**(*title* : str, *fitdlgfunc* : Callable) → None  
 Compute fitting curve using an interactive dialog  
**Paramètres**  
 — **title** – Title of the dialog  
 — **fitdlgfunc** – Fitting dialog function

**compute\_multigaussianfit**() → None  
 Compute multi-Gaussian fitting curve using an interactive dialog

**compute\_fwhm**(*param* : FWHMParam | None = None) → dict[str, ResultShape]  
 Compute FWHM with `cdl.computation.signal.compute_fwhm()`

**compute\_fw1e2**() → dict[str, ResultShape]  
 Compute FW at  $1/e^2$  with `cdl.computation.signal.compute_fw1e2()`

**compute\_stats**() → dict[str, ResultProperties]  
 Compute data statistics with `cdl.computation.signal.compute_stats()`

**compute\_histogram**(*param* : HistogramParam | None = None) → dict[str, ResultShape]  
Compute histogram with `cdl.computation.signal.compute_histogram()`

**compute\_contrast**() → dict[str, ResultProperties]  
Compute contrast with `cdl.computation.signal.compute_contrast()`

**compute\_x\_at\_minmax**() → dict[str, ResultProperties]  
Compute x at min/max with `cdl.computation.signal.compute_x_at_minmax()`

**compute\_sampling\_rate\_period**() → dict[str, ResultProperties]  
Compute sampling rate and period (mean and std) with `cdl.computation.signal.compute_sampling_rate_period()`

**compute\_bandwidth\_3db**() → None  
Compute bandwidth at -3dB with `cdl.computation.signal.compute_bandwidth_3db()`

**compute\_dynamic\_parameters**(*param* : DynamicParam | None = None) → dict[str, ResultProperties]  
Compute Dynamic Parameters (ENOB, SINAD, THD, SFDR, SNR) with `cdl.computation.signal.compute_dynamic_parameters()`

## Processeur d'image

Lorsque vous travaillez avec des images, les méthodes de `cdl.core.gui.processor.image.ImageProcessor` peuvent être accédées.

**class** `cdl.core.gui.processor.image.ImageProcessor`(*panel* : SignalPanel | ImagePanel, *plotwidget* : PlotWidget)

Objet de traitement d'image : opérations, traitement, analyse

**compute\_normalize**(*param* : NormalizeParam | None = None) → None  
Normalize data with `cdl.computation.image.compute_normalize()`

**compute\_sum**() → None  
Compute sum with `cdl.computation.image.compute_addition()`

**compute\_addition\_constant**(*param* : ConstantParam | None = None) → None  
Compute sum with a constant using `cdl.computation.image.compute_addition_constant()`

**compute\_average**() → None  
Compute average with `cdl.computation.image.compute_addition()` and dividing by the number of images

**compute\_product**() → None  
Compute product with `cdl.computation.image.compute_product()`

**compute\_product\_constant**(*param* : ConstantParam | None = None) → None  
Compute product with a constant using `cdl.computation.image.compute_product_constant()`

**compute\_logp1**(*param* : LogP1Param | None = None) → None  
Compute base 10 logarithm using `cdl.computation.image.compute_logp1()`

**compute\_rotate**(*param* : RotateParam | None = None) → None  
Rotate data arbitrarily using `cdl.computation.image.compute_rotate()`

**compute\_rotate90**() → None  
Rotate data 90° with `cdl.computation.image.compute_rotate90()`

**compute\_rotate270()** → None  
 Rotate data 270° with `cdl.computation.image.compute_rotate270()`

**compute\_fliph()** → None  
 Flip data horizontally using `cdl.computation.image.compute_fliph()`

**compute\_flipv()** → None  
 Flip data vertically with `cdl.computation.image.compute_flipv()`

**distribute\_on\_grid**(*param* : GridParam | None = None) → None  
 Distribuer les images sur une grille

**reset\_positions()** → None  
 Réinitialiser les positions des images

**compute\_resize**(*param* : ResizeParam | None = None) → None  
 Resize image with `cdl.computation.image.compute_resize()`

**compute\_binning**(*param* : BinningParam | None = None) → None  
 Binning image with `cdl.computation.image.compute_binning()`

**compute\_line\_profile**(*param* : LineProfileParam | None = None) → None  
 Compute profile along a vertical or horizontal line with `cdl.computation.image.compute_line_profile()`

**compute\_segment\_profile**(*param* : SegmentProfileParam | None = None)  
 Compute profile along a segment with `cdl.computation.image.compute_segment_profile()`

**compute\_average\_profile**(*param* : AverageProfileParam | None = None) → None  
 Compute average profile with `cdl.computation.image.compute_average_profile()`

**compute\_radial\_profile**(*param* : RadialProfileParam | None = None) → None  
 Compute radial profile with `cdl.computation.image.compute_radial_profile()`

**compute\_histogram**(*param* : HistogramParam | None = None) → None  
 Compute histogram with `cdl.computation.image.compute_histogram()`

**compute\_swap\_axes()** → None  
 Swap data axes with `cdl.computation.image.compute_swap_axes()`

**compute\_abs()** → None  
 Compute absolute value with `cdl.computation.image.compute_abs()`

**compute\_re()** → None  
 Compute real part with `cdl.computation.image.compute_re()`

**compute\_im()** → None  
 Compute imaginary part with `cdl.computation.image.compute_im()`

**compute\_astype**(*param* : DataTypeIParm | None = None) → None  
 Convert data type with `cdl.computation.image.compute_astype()`

**compute\_log10()** → None  
 Compute Log10 with `cdl.computation.image.compute_log10()`

**compute\_exp()** → None  
 Compute Log10 with `cdl.computation.image.compute_exp()`

**compute\_arithmetic**(obj2 : ImageObj | None = None, param : ArithmeticParam | None = None) → None  
Compute arithmetic operation between two images with `cdl.computation.image.compute_arithmetic()`

**compute\_difference**(obj2 : ImageObj | list[ImageObj] | None = None) → None  
Compute difference between two images with `cdl.computation.image.compute_difference()`

**compute\_difference\_constant**(param : ConstantParam | None = None) → None  
Compute difference with a constant with `cdl.computation.image.compute_difference_constant()`

**compute\_quadratic\_difference**(obj2 : ImageObj | list[ImageObj] | None = None) → None  
Compute quadratic difference between two images with `cdl.computation.image.compute_quadratic_difference()`

**compute\_division**(obj2 : ImageObj | list[ImageObj] | None = None) → None  
Compute division between two images with `cdl.computation.image.compute_division()`

**compute\_division\_constant**(param : ConstantParam | None = None) → None  
Compute division by a constant with `cdl.computation.image.compute_division_constant()`

**compute\_flatfield**(obj2 : ImageObj | None = None, param : FlatFieldParam | None = None) → None  
Compute flat field correction with `cdl.computation.image.compute_flatfield()`

**compute\_calibration**(param : ZCalibrateParam | None = None) → None  
Compute data linear calibration with `cdl.computation.image.compute_calibration()`

**compute\_clip**(param : ClipParam | None = None) → None  
Compute maximum data clipping with `cdl.computation.image.compute_clip()`

**compute\_offset\_correction**(param : ROI2DParam | None = None) → None  
Compute offset correction with `cdl.computation.image.compute_offset_correction()`

**compute\_gaussian\_filter**(param : GaussianParam | None = None) → None  
Compute gaussian filter with `cdl.computation.image.compute_gaussian_filter()`

**compute\_moving\_average**(param : MovingAverageParam | None = None) → None  
Compute moving average with `cdl.computation.image.compute_moving_average()`

**compute\_moving\_median**(param : MovingMedianParam | None = None) → None  
Compute moving median with `cdl.computation.image.compute_moving_median()`

**compute\_wiener**() → None  
Compute Wiener filter with `cdl.computation.image.compute_wiener()`

**compute\_fft**(param : FFTParam | None = None) → None  
Compute FFT with `cdl.computation.image.compute_fft()`

**compute\_ifft**(param : FFTParam | None = None) → None  
Compute iFFT with `cdl.computation.image.compute_ifft()`

**compute\_magnitude\_spectrum**(param : SpectrumParam | None = None) → None  
Compute magnitude spectrum with `cdl.computation.image.compute_magnitude_spectrum()`

**compute\_phase\_spectrum**() → None  
Compute phase spectrum with `cdl.computation.image.compute_phase_spectrum()`

**compute\_psd**(param : SpectrumParam | None = None) → None  
Compute Power Spectral Density (PSD) with `cdl.computation.image.compute_psd()`

**compute\_butterworth**(*param* : ButterworthParam | *None* = *None*) → *None*  
 Compute Butterworth filter with `cdl.computation.image.compute_butterworth()`

**compute\_threshold**(*param* : ThresholdParam | *None* = *None*) → *None*  
 Compute parametric threshold with `cdl.computation.image.threshold.compute_threshold()`

**compute\_threshold\_isodata**() → *None*  
 Compute threshold using Isodata algorithm with `cdl.computation.image.threshold.compute_threshold_isodata()`

**compute\_threshold\_li**() → *None*  
 Compute threshold using Li algorithm with `cdl.computation.image.threshold.compute_threshold_li()`

**compute\_threshold\_mean**() → *None*  
 Compute threshold using Mean algorithm with `cdl.computation.image.threshold.compute_threshold_mean()`

**compute\_threshold\_minimum**() → *None*  
 Compute threshold using Minimum algorithm with `cdl.computation.image.threshold.compute_threshold_minimum()`

**compute\_threshold\_otsu**() → *None*  
 Compute threshold using Otsu algorithm with `cdl.computation.image.threshold.compute_threshold_otsu()`

**compute\_threshold\_triangle**() → *None*  
 Compute threshold using Triangle algorithm with `cdl.computation.image.threshold.compute_threshold_triangle()`

**compute\_threshold\_yen**() → *None*  
 Compute threshold using Yen algorithm with `cdl.computation.image.threshold.compute_threshold_yen()`

**compute\_all\_threshold**() → *None*  
 Compute all threshold algorithms using the following functions :  
 — `cdl.computation.image.threshold.compute_threshold_isodata()`  
 — `cdl.computation.image.threshold.compute_threshold_li()`  
 — `cdl.computation.image.threshold.compute_threshold_mean()`  
 — `cdl.computation.image.threshold.compute_threshold_minimum()`  
 — `cdl.computation.image.threshold.compute_threshold_otsu()`  
 — `cdl.computation.image.threshold.compute_threshold_triangle()`  
 — `cdl.computation.image.threshold.compute_threshold_yen()`

**compute\_adjust\_gamma**(*param* : AdjustGammaParam | *None* = *None*) → *None*  
 Compute gamma correction with `cdl.computation.image.exposure.compute_adjust_gamma()`

**compute\_adjust\_log**(*param* : AdjustLogParam | *None* = *None*) → *None*  
 Compute log correction with `cdl.computation.image.exposure.compute_adjust_log()`

**compute\_adjust\_sigmoid**(*param* : AdjustSigmoidParam | *None* = *None*) → *None*  
 Compute sigmoid correction with `cdl.computation.image.exposure.compute_adjust_sigmoid()`

**compute\_rescale\_intensity**(*param* : RescaleIntensityParam | *None* = *None*) → *None*  
 Rescale image intensity levels with `:py:func`cdl.computation.image.exposure.compute_rescale_intensity``

```

compute_equalize_hist(param : EqualizeHistParam | None = None) → None
    Histogram equalization with cdl.computation.image.exposure.compute_equalize_hist()

compute_equalize_adapthist(param : EqualizeAdaptHistParam | None = None) → None
    Adaptive histogram equalization with cdl.computation.image.exposure.compute_equalize_adapthist()

compute_denoise_tv(param : DenoiseTVParam | None = None) → None
    Compute Total Variation denoising with cdl.computation.image.restoration.compute_denoise_tv()

compute_denoise_bilateral(param : DenoiseBilateralParam | None = None) → None
    Compute bilateral filter denoising with cdl.computation.image.restoration.compute_denoise_bilateral()

compute_denoise_wavelet(param : DenoiseWaveletParam | None = None) → None
    Compute Wavelet denoising with cdl.computation.image.restoration.compute_denoise_wavelet()

compute_denoise_tophat(param : MorphologyParam | None = None) → None
    Denoise using White Top-Hat with cdl.computation.image.restoration.compute_denoise_tophat()

compute_all_denoise(params : list | None = None) → None
    Compute all denoising filters using the following functions :
    — cdl.computation.image.restoration.compute_denoise_tv()
    — cdl.computation.image.restoration.compute_denoise_bilateral()
    — cdl.computation.image.restoration.compute_denoise_wavelet()
    — cdl.computation.image.restoration.compute_denoise_tophat()

compute_white_tophat(param : MorphologyParam | None = None) → None
    Compute White Top-Hat with cdl.computation.image.morphology.compute_white_tophat()

compute_black_tophat(param : MorphologyParam | None = None) → None
    Compute Black Top-Hat with cdl.computation.image.morphology.compute_black_tophat()

compute_erosion(param : MorphologyParam | None = None) → None
    Compute Erosion with cdl.computation.image.morphology.compute_erosion()

compute_dilation(param : MorphologyParam | None = None) → None
    Compute Dilation with cdl.computation.image.morphology.compute_dilation()

compute_opening(param : MorphologyParam | None = None) → None
    Compute morphological opening with cdl.computation.image.morphology.compute_opening()

compute_closing(param : MorphologyParam | None = None) → None
    Compute morphological closing with cdl.computation.image.morphology.compute_closing()

compute_all_morphology(param : MorphologyParam | None = None) → None
    Compute all morphology filters using the following functions :
    — cdl.computation.image.morphology.compute_white_tophat()
    — cdl.computation.image.morphology.compute_black_tophat()
    — cdl.computation.image.morphology.compute_erosion()
    — cdl.computation.image.morphology.compute_dilation()
    — cdl.computation.image.morphology.compute_opening()
    — cdl.computation.image.morphology.compute_closing()

```

**compute\_canny**(*param* : CannyParam | None = None) → None  
 Compute Canny filter with `cdl.computation.image.edges.compute_canny()`

**compute\_roberts**() → None  
 Compute Roberts filter with `cdl.computation.image.edges.compute_roberts()`

**compute\_prewitt**() → None  
 Compute Prewitt filter with `cdl.computation.image.edges.compute_prewitt()`

**compute\_prewitt\_h**() → None  
 Compute Prewitt filter (horizontal) with `cdl.computation.image.edges.compute_prewitt_h()`

**compute\_prewitt\_v**() → None  
 Compute Prewitt filter (vertical) with `cdl.computation.image.edges.compute_prewitt_v()`

**compute\_sobel**() → None  
 Compute Sobel filter with `cdl.computation.image.edges.compute_sobel()`

**compute\_sobel\_h**() → None  
 Compute Sobel filter (horizontal) with `cdl.computation.image.edges.compute_sobel_h()`

**compute\_sobel\_v**() → None  
 Compute Sobel filter (vertical) with `cdl.computation.image.edges.compute_sobel_v()`

**compute\_scharr**() → None  
 Compute Scharr filter with `cdl.computation.image.edges.compute_scharr()`

**compute\_scharr\_h**() → None  
 Compute Scharr filter (horizontal) with `cdl.computation.image.edges.compute_scharr_h()`

**compute\_scharr\_v**() → None  
 Compute Scharr filter (vertical) with `cdl.computation.image.edges.compute_scharr_v()`

**compute\_farid**() → None  
 Compute Farid filter with `cdl.computation.image.edges.compute_farid()`

**compute\_farid\_h**() → None  
 Compute Farid filter (horizontal) with `cdl.computation.image.edges.compute_farid_h()`

**compute\_farid\_v**() → None  
 Compute Farid filter (vertical) with `cdl.computation.image.edges.compute_farid_v()`

**compute\_laplace**() → None  
 Compute Laplace filter with `cdl.computation.image.edges.compute_laplace()`

**compute\_all\_edges**() → None  
 Compute all edges filters using the following functions :  
 — `cdl.computation.image.edges.compute_roberts()`  
 — `cdl.computation.image.edges.compute_prewitt()`  
 — `cdl.computation.image.edges.compute_prewitt_h()`  
 — `cdl.computation.image.edges.compute_prewitt_v()`  
 — `cdl.computation.image.edges.compute_sobel()`  
 — `cdl.computation.image.edges.compute_sobel_h()`  
 — `cdl.computation.image.edges.compute_sobel_v()`  
 — `cdl.computation.image.edges.compute_scharr()`  
 — `cdl.computation.image.edges.compute_scharr_h()`  
 — `cdl.computation.image.edges.compute_scharr_v()`  
 — `cdl.computation.image.edges.compute_farid()`

- `cdl.computation.image.edges.compute_farid_h()`
- `cdl.computation.image.edges.compute_farid_v()`
- `cdl.computation.image.edges.compute_laplace()`

**compute\_stats()** → dict[str, ResultProperties]  
Compute data statistics with `cdl.computation.image.compute_stats()`

**compute\_centroid()** → dict[str, ResultShape]  
Compute image centroid with `cdl.computation.image.compute_centroid()`

**compute\_enclosing\_circle()** → dict[str, ResultShape]  
Compute minimum enclosing circle with `cdl.computation.image.compute_enclosing_circle()`

**compute\_peak\_detection**(param : Peak2DDetectionParam | None = None) → dict[str, ResultShape]  
Compute 2D peak detection with `cdl.computation.image.compute_peak_detection()`

**compute\_contour\_shape**(param : ContourShapeParam | None = None) → dict[str, ResultShape]  
Compute contour shape fit with `cdl.computation.image.detection.compute_contour_shape()`

**compute\_hough\_circle\_peaks**(param : HoughCircleParam | None = None) → dict[str, ResultShape]  
Compute peak detection based on a circle Hough transform with `cdl.computation.image.compute_hough_circle_peaks()`

**compute\_blob\_dog**(param : BlobDOGParam | None = None) → dict[str, ResultShape]  
Compute blob detection using Difference of Gaussian method with `cdl.computation.image.detection.compute_blob_dog()`

**compute\_blob\_doh**(param : BlobDOHParam | None = None) → dict[str, ResultShape]  
Compute blob detection using Determinant of Hessian method with `cdl.computation.image.detection.compute_blob_doh()`

**compute\_blob\_log**(param : BlobLOGParam | None = None) → dict[str, ResultShape]  
Compute blob detection using Laplacian of Gaussian method with `cdl.computation.image.detection.compute_blob_log()`

**compute\_blob\_opencv**(param : BlobOpenCVParam | None = None) → dict[str, ResultShape]  
Compute blob detection using OpenCV with `cdl.computation.image.detection.compute_blob_opencv()`

## 2.2.5 Modèle de données interne

Dans son modèle de données interne, DataLab stocke les données à l'aide de deux classes principales :

- `cdl.obj.SignalObj`, qui représente un objet signal, et
- `cdl.obj.ImageObj`, qui représente un objet image.

Ces classes sont définies dans le paquet `cdl.core.model` mais sont exposées publiquement dans le paquet `cdl.obj`.

Par ailleurs, DataLab utilise de nombreux jeux de données différents (basés sur la classe `DataSet` de `guidata`) pour stocker les paramètres des calculs. Ces jeux de données sont définis dans différents modules mais sont exposés publiquement dans le paquet `cdl.param`.

**Voir aussi :**

La section [API](#) pour plus d'informations sur l'API publique.

## 2.2.6 Plugins

DataLab est une application modulaire. Il est possible d'ajouter de nouvelles fonctionnalités à DataLab en écrivant des plugins. Un plugin est un module Python qui est chargé au démarrage par DataLab. Un plugin peut ajouter de nouvelles fonctionnalités à DataLab, ou modifier des fonctionnalités existantes.

Le système de plugins prend actuellement en charge les fonctionnalités suivantes :

- Fonctionnalités de traitement : ajouter de nouvelles tâches de traitement au système de traitement DataLab, y compris des interfaces graphiques spécifiques.
- Entrée/sortie : ajouter de nouveaux formats de fichiers au système d'entrée/sortie de DataLab.
- Fonctionnalités HDF5 : ajouter de nouveaux formats de fichiers HDF5 au système d'entrée/sortie HDF5 de DataLab.

### Qu'est-ce qu'un plugin ?

Un plugin est un module Python qui est chargé au démarrage par DataLab. Un plugin peut ajouter de nouvelles fonctionnalités à DataLab, ou modifier des fonctionnalités existantes.

Un plugin est un module Python dont le nom de fichier commence par `cdl_` et qui contient une classe dérivée de la classe `cdl.plugins.PluginBase`. Le nom de la classe n'est pas important, tant qu'elle est dérivée de `cdl.plugins.PluginBase` et qu'elle dispose d'un attribut `PLUGIN_INFO` qui est une instance de la classe `cdl.plugins.PluginInfo`. L'attribut `PLUGIN_INFO` est utilisé par DataLab pour récupérer des informations sur le plugin.

---

**Note :** Le mécanisme de découverte de plugins de DataLab ne chargera que les plugins définis dans des fichiers Python dont les noms commencent par `cdl_` (par exemple `cdl_monplugin.py`).

---

### Où est positionné un plugin ?

Etant donné que les plugins sont des modules Python, ils peuvent être placés n'importe où dans le chemin Python de l'installation de DataLab.

Des emplacements supplémentaires spéciaux sont disponibles pour les plugins :

- Le répertoire *plugins* dans le dossier de configuration de l'utilisateur (par exemple `C:/Users/JohnDoe/.DataLab/plugins` sur Windows ou `~/.DataLab/plugins` sur Linux).
- Le répertoire *plugins* dans le même dossier que l'exécutable *DataLab* en cas d'installation autonome.
- Le répertoire *plugins* dans le package *cdl* en cas de plugins internes uniquement (c'est-à-dire qu'il n'est pas recommandé d'y placer vos propres plugins).

### Comment développer un plugin ?

Pour développer un plugin, vous pouvez commencer par dériver de l'un des exemples de plugins (voir ci-dessous) et le modifier pour répondre à vos besoins.

Si vous souhaitez coder un plugin dans votre environnement de développement Python habituel (de préférence avec un IDE comme *Spyder*) et profiter de la complétion de code, vous pouvez ajouter le package *cdl* à votre chemin Python.

Cela peut être fait :

- En installant DataLab dans votre environnement Python (en utilisant l'une des méthodes suivantes : *Paquet Conda*, *Gestionnaire de paquets pip*, *Paquet Wheel*, ou *Paquet source*),
- Ou en ajoutant le package *cdl* à votre chemin Python manuellement :
  - Téléchargez le code source de DataLab depuis la page *PyPI*,
  - Décompressez le code source dans un dossier sur votre ordinateur,

- Ajoutez le package *cdl* à votre chemin Python (par exemple en utilisant le *Gestionnaire de PYTHONPATH* dans Spyder).

---

**Note :** Même si vous avez correctement installé le package *cdl* dans votre environnement Python, vous ne pourrez pas exécuter l'application DataLab depuis votre environnement de développement pour tester votre plugin. Vous devrez exécuter DataLab à partir de la ligne de commande ou du raccourci créé lors de l'installation.

---

### Exemple : plugin de traitement

Voici un exemple simple d'un plugin qui ajoute une nouvelle fonctionnalité à DataLab.

```
# Copyright (c) DataLab Platform Developers, BSD 3-Clause license, see LICENSE file.

"""
Test Data Plugin for DataLab
-----

This plugin is an example of DataLab plugin. It provides test data samples
and some actions to test DataLab functionalities.
"""

import cdl.obj as dlo
import cdl.tests.data as test_data
from cdl.computation import image as cpima
from cdl.computation import signal as cpsig
from cdl.config import _
from cdl.plugins import PluginBase, PluginInfo

# -----
# All computation functions must be defined as global functions, otherwise
# they cannot be pickled and sent to the worker process
# -----

def add_noise_to_signal(
    src: dlo.SignalObj, p: test_data.GaussianNoiseParam
) -> dlo.SignalObj:
    """Add gaussian noise to signal"""
    dst = cpsig.dst_l1(src, "add_gaussian_noise", f"mu={p.mu},sigma={p.sigma}")
    test_data.add_gaussian_noise_to_signal(dst, p)
    return dst

def add_noise_to_image(src: dlo.ImageObj, p: dlo.NormalRandomParam) -> dlo.ImageObj:
    """Add gaussian noise to image"""
    dst = cpima.dst_l1(src, "add_gaussian_noise", f"mu={p.mu},sigma={p.sigma}")
    test_data.add_gaussian_noise_to_image(dst, p)
    return dst

class PluginTestData(PluginBase):
    """DataLab Test Data Plugin"""
```

(suite sur la page suivante)

(suite de la page précédente)

```

PLUGIN_INFO = PluginInfo(
    name=_("Test data"),
    version="1.0.0",
    description=_("Testing DataLab functionalities"),
)

# Signal processing features -----
def add_noise_to_signal(self) -> None:
    """Add noise to signal"""
    self.signalpanel.processor.compute_11(
        add_noise_to_signal,
        paramclass=test_data.GaussianNoiseParam,
        title=_("Add noise"),
    )

def create_paracetamol_signal(self) -> None:
    """Create paracetamol signal"""
    obj = test_data.create_paracetamol_signal()
    self.proxy.add_object(obj)

def create_noisy_signal(self) -> None:
    """Create noisy signal"""
    obj = self.signalpanel.new_object(add_to_panel=False)
    if obj is not None:
        noiseparam = test_data.GaussianNoiseParam(_("Noise"))
        self.signalpanel.processor.update_param_defaults(noiseparam)
        if noiseparam.edit(self.main):
            test_data.add_gaussian_noise_to_signal(obj, noiseparam)
            self.proxy.add_object(obj)

# Image processing features -----
def add_noise_to_image(self) -> None:
    """Add noise to image"""
    self.imagepanel.processor.compute_11(
        add_noise_to_image,
        paramclass=dlo.NormalRandomParam,
        title=_("Add noise"),
    )

def create_peak2d_image(self) -> None:
    """Create 2D peak image"""
    obj = self.imagepanel.new_object(add_to_panel=False)
    if obj is not None:
        param = test_data.PeakDataParam.create(size=max(obj.data.shape))
        self.imagepanel.processor.update_param_defaults(param)
        if param.edit(self.main):
            obj.data = test_data.get_peak2d_data(param)
            self.proxy.add_object(obj)

def create_sincos_image(self) -> None:
    """Create 2D sin cos image"""

```

(suite sur la page suivante)

(suite de la page précédente)

```

newparam = self.edit_new_image_parameters(hide_image_type=True)
if newparam is not None:
    obj = test_data.create_sincos_image(newparam)
    self.proxy.add_object(obj)

def create_noisygauss_image(self) -> None:
    """Create 2D noisy gauss image"""
    newparam = self.edit_new_image_parameters(hide_image_type=True)
    if newparam is not None:
        obj = test_data.create_noisygauss_image(newparam, add_annotations=False)
        self.proxy.add_object(obj)

def create_multigauss_image(self) -> None:
    """Create 2D multi gauss image"""
    newparam = self.edit_new_image_parameters(hide_image_type=True)
    if newparam is not None:
        obj = test_data.create_multigauss_image(newparam)
        self.proxy.add_object(obj)

def create_2dstep_image(self) -> None:
    """Create 2D step image"""
    newparam = self.edit_new_image_parameters(hide_image_type=True)
    if newparam is not None:
        obj = test_data.create_2dstep_image(newparam)
        self.proxy.add_object(obj)

def create_ring_image(self) -> None:
    """Create 2D ring image"""
    param = test_data.RingParam(_("Ring"))
    if param.edit(self.main):
        obj = test_data.create_ring_image(param)
        self.proxy.add_object(obj)

def create_annotated_image(self) -> None:
    """Create annotated image"""
    obj = test_data.create_annotated_image()
    self.proxy.add_object(obj)

# Plugin menu entries -----
def create_actions(self) -> None:
    """Create actions"""
    # Signal Panel -----
    sah = self.signalpanel.acthandler
    with sah.new_menu(_("Test data")):
        sah.new_action(_("Add noise to signal"), triggered=self.add_noise_to_signal)
        sah.new_action(
            _("Load spectrum of paracetamol"),
            triggered=self.create_paracetamol_signal,
            select_condition="always",
            separator=True,
        )
        sah.new_action(

```

(suite sur la page suivante)

(suite de la page précédente)

```

        _("Create noisy signal"),
        triggered=self.create_noisy_signal,
        select_condition="always",
    )
# Image Panel -----
iah = self.imagepanel.acthandler
with iah.new_menu(_("Test data")):
    iah.new_action(_("Add noise to image"), triggered=self.add_noise_to_image)
    # with iah.new_menu(_("Data samples")):
    iah.new_action(
        _("Create image with peaks"),
        triggered=self.create_peak2d_image,
        select_condition="always",
        separator=True,
    )
    iah.new_action(
        _("Create 2D sin cos image"),
        triggered=self.create_sincos_image,
        select_condition="always",
    )
    iah.new_action(
        _("Create 2D noisy gauss image"),
        triggered=self.create_noisygauss_image,
        select_condition="always",
    )
    iah.new_action(
        _("Create 2D multi gauss image"),
        triggered=self.create_multigauss_image,
        select_condition="always",
    )
    iah.new_action(
        _("Create annotated image"),
        triggered=self.create_annotated_image,
        select_condition="always",
    )
    iah.new_action(
        _("Create 2D step image"),
        triggered=self.create_2dstep_image,
        select_condition="always",
    )
    iah.new_action(
        _("Create ring image"),
        triggered=self.create_ring_image,
        select_condition="always",
    )

```

**Exemple : plugin d'entrée/sortie**

Voici un exemple simple d'un plugin qui ajoute de nouveaux formats de fichiers à DataLab.

```
# Copyright (c) DataLab Platform Developers, BSD 3-Clause license, see LICENSE file.

"""
Image file formats Plugin for DataLab
-----

This plugin is an example of DataLab plugin.
It provides image file formats from cameras, scanners, and other acquisition devices.
"""

import struct

import numpy as np

from cdl.core.io.base import FormatInfo
from cdl.core.io.image.base import ImageFormatBase

# =====
# Thales Pixium FXD file format
# =====

class FXDFile:
    """Class implementing Thales Pixium FXD Image file reading feature

    Args:
        fname (str): path to FXD file
        debug (bool): debug mode
    """

    HEADER = "<llllllffl"

    def __init__(self, fname: str = None, debug: bool = False) -> None:
        self.__debug = debug
        self.file_format = None # long
        self.nbcols = None # long
        self.nbrows = None # long
        self.nbframes = None # long
        self.pixeltype = None # long
        self.quantlevels = None # long
        self.maxlevel = None # float
        self.minlevel = None # float
        self.comment_length = None # long
        self.fname = None
        self.data = None
        if fname is not None:
            self.load(fname)

    def __repr__(self) -> str:
```

(suite sur la page suivante)

(suite de la page précédente)

```

"""Return a string representation of the object"""
info = (
    ("Image width", f"{self.nbcolls:d}"),
    ("Image Height", f"{self.nbrows:d}"),
    ("Frame number", f"{self.nbframes:d}"),
    ("File format", f"{self.file_format:d}"),
    ("Pixel type", f"{self.pixeltype:d}"),
    ("Quantlevels", f"{self.quantlevels:d}"),
    ("Min. level", f"{self.minlevel:f}"),
    ("Max. level", f"{self.maxlevel:f}"),
    ("Comment length", f"{self.comment_length:d}"),
)
desc_len = max(len(d) for d in list(zip(*info))[0]) + 3
res = ""
for description, value in info:
    res += ("{: " + str(desc_len) + "}}{}\n").format(description + ": ", value)

res = object.__repr__(self) + "\n" + res
return res

def load(self, fname: str) -> None:
    """Load header and image pixel data

    Args:
        fname (str): path to FXD file
    """
```

```

    with open(fname, "rb") as data_file:
        header_s = struct.Struct(self.HEADER)
        record = data_file.read(9 * 4)
        unpacked_rec = header_s.unpack(record)
        (
            self.file_format,
            self.nbcolls,
            self.nbrows,
            self.nbframes,
            self.pixeltype,
            self.quantlevels,
            self.maxlevel,
            self.minlevel,
            self.comment_length,
        ) = unpacked_rec
        if self.__debug:
            print(unpacked_rec)
            print(self)
        data_file.seek(128 + self.comment_length)
        if self.pixeltype == 0:
            size, dtype = 4, np.float32
        elif self.pixeltype == 1:
            size, dtype = 2, np.uint16
        elif self.pixeltype == 2:
            size, dtype = 1, np.uint8
        else:

```

(suite sur la page suivante)

(suite de la page précédente)

```

        raise NotImplementedError(f"Unsupported pixel type: {self.pixeltype}")
    block = data_file.read(self.nbrows * self.nbcolls * size)
    data = np.frombuffer(block, dtype=dtype)
    self.data = data.reshape(self.nbrows, self.nbcolls)

class FXDImageFormat(ImageFormatBase):
    """Object representing Thales Pixium (FXD) image file type"""

    FORMAT_INFO = FormatInfo(
        name="Thales Pixium",
        extensions="*.fxd",
        readable=True,
        writeable=False,
    )

    @staticmethod
    def read_data(filename: str) -> np.ndarray:
        """Read data and return it

        Args:
            filename (str): path to FXD file

        Returns:
            np.ndarray: image data
        """
        fxd_file = FXDFile(filename)
        return fxd_file.data

# =====
# Dürr NDT XYZ file format
# =====

class XYZImageFormat(ImageFormatBase):
    """Object representing Dürr NDT XYZ image file type"""

    FORMAT_INFO = FormatInfo(
        name="Dürr NDT",
        extensions="*.xyz",
        readable=True,
        writeable=True,
    )

    @staticmethod
    def read_data(filename: str) -> np.ndarray:
        """Read data and return it

        Args:
            filename (str): path to XYZ file

```

(suite sur la page suivante)

(suite de la page précédente)

```

Returns:
    np.ndarray: image data
    """
    with open(filename, "rb") as fdesc:
        cols = int(np.fromfile(fdesc, dtype=np.uint16, count=1)[0])
        rows = int(np.fromfile(fdesc, dtype=np.uint16, count=1)[0])
        arr = np.fromfile(fdesc, dtype=np.uint16, count=cols * rows)
        arr = arr.reshape((rows, cols))
    return np.fliplr(arr)

@staticmethod
def write_data(filename: str, data: np.ndarray) -> None:
    """Write data to file

    Args:
        filename: File name
        data: Image array data
    """
    data = np.fliplr(data)
    with open(filename, "wb") as fdesc:
        fdesc.write(np.array(data.shape[1], dtype=np.uint16).tobytes())
        fdesc.write(np.array(data.shape[0], dtype=np.uint16).tobytes())
        fdesc.write(data.tobytes())

```

## Autres exemples

D'autres exemples de plugins peuvent être trouvés dans le répertoire *plugins/examples* du code source de DataLab (explorez [ici](#) sur [GitHub](#)).

## API publique

### Système de plugins de DataLab

Le système de plugins de DataLab fournit un moyen d'étendre l'application avec de nouvelles fonctionnalités.

Les plugins sont des modules Python qui reposent sur deux classes :

- *PluginInfo*, qui stocke des informations sur le plugin
- *PluginBase*, qui est la classe de base pour tous les plugins

Les plugins peuvent également étendre les fonctionnalités d'entrée/sortie de DataLab en fournissant de nouveaux formats d'image ou de signal. Pour ce faire, ils doivent fournir une sous-classe de *ImageFormatBase* ou *SignalFormatBase*, dans laquelle les informations de format sont définies à l'aide de la classe *FormatInfo*.

```
class cdl.plugins.PluginRegistry(name, bases, attrs)
```

Métaclasse pour l'enregistrement des plugins

```
classmethod get_plugin_classes() -> list[type[PluginBase]]
```

Retourne les classes de plugins

```
classmethod get_plugins() -> list[PluginBase]
```

Retourne les instances de plugins

**classmethod** `get_plugin(name_or_class : str | type[PluginBase]) → PluginBase | None`

Retourne l'instance de plugin

**classmethod** `register_plugin(plugin : PluginBase)`

Enregistrer le plugin

**classmethod** `unregister_plugin(plugin : PluginBase)`

Désenregistrer le plugin

**classmethod** `get_plugin_infos(html : bool = True) → str`

Retourne les informations sur les plugins (noms, versions, descriptions) au format html

**Paramètres**

**html** – retourner du texte formaté en html (par défaut : True)

**class** `cdl.plugins.PluginInfo(name : str = None, version : str = '0.0.0', description : str = '', icon : str = None)`

Informations sur le plugin

**class** `cdl.plugins.PluginBaseMeta(name, bases, namespace, /, **kwargs)`

Métaclasse mixte pour éviter les conflits

**class** `cdl.plugins.PluginBase`

Classe de base du plugin

**property** `signalpanel : SignalPanel`

Retourne le panneau de signal

**property** `imagepanel : ImagePanel`

Retourne le panneau d'image

**show\_warning**(message : str)

Afficher un message d'avertissement

**show\_error**(message : str)

Afficher un message d'erreur

**show\_info**(message : str)

Afficher un message d'information

**ask\_yesno**(message : str, title : str | None = None, cancelable : bool = False) → bool

Poser une question oui/non

**edit\_new\_signal\_parameters**(title : str | None = None, size : int | None = None, hide\_signal\_type : bool = True) → NewSignalParam

Créer et éditer un nouveau jeu de paramètres de signal

**Paramètres**

— **title** – titre du nouveau signal

— **size** – taille du nouveau signal (par défaut : None, obtenue à partir du signal actuel)

— **hide\_signal\_type** – masquer le paramètre de type de signal (par défaut : True)

**Renvoie**

Nouveau jeu de paramètres de signal (ou None si annulé)

**edit\_new\_image\_parameters**(title : str | None = None, shape : tuple[int, int] | None = None, hide\_image\_type : bool = True, hide\_image\_dtype : bool = False) → NewImageParam | None

Créer et éditer un nouveau jeu de paramètres d'image

**Paramètres**

- **title** – titre de la nouvelle image
- **shape** – dimensions de la nouvelle image (par défaut : None, obtenues à partir de l'image actuelle)
- **hide\_image\_type** – masquer le paramètre de type d'image (par défaut : True)
- **hide\_image\_dtype** – masquer le paramètre de type de données d'image (par défaut : False)

**Renvoie**

Nouveau jeu de paramètres d'image (ou None si annulé)

**is\_registered()**

Retourne True si le plugin est enregistré

**register(main : [main.CDLMainWindow](#)) → None**

Enregistrer le plugin

**unregister()**

Désenregistrer le plugin

**register\_hooks()**

Enregistrer les hooks du plugin

**unregister\_hooks()**

Désenregistrer les hooks du plugin

**abstract create\_actions()**

Créer des actions

**cdl.plugins.discover\_plugins() → list[type[[PluginBase](#)]]**

Découvrir les plugins en utilisant la convention de nommage

**Renvoie**

Liste des plugins découverts (en tant que classes)

**cdl.plugins.get\_available\_plugins() → list[[PluginBase](#)]**

Instancier et obtenir les plugins disponibles

**Renvoie**

Liste des plugins disponibles (en tant qu'instances)

## 2.2.7 Journaux de bord

Malgré des efforts considérables en matière de tests (tests unitaires, couverture de tests, etc.), DataLab peut s'arrêter inopinément ou se comporter de façon inattendue.

**Pour traiter ce type de situation, DataLab fournit deux types de journaux de bord (localisés dans votre répertoire utilisateur) :**

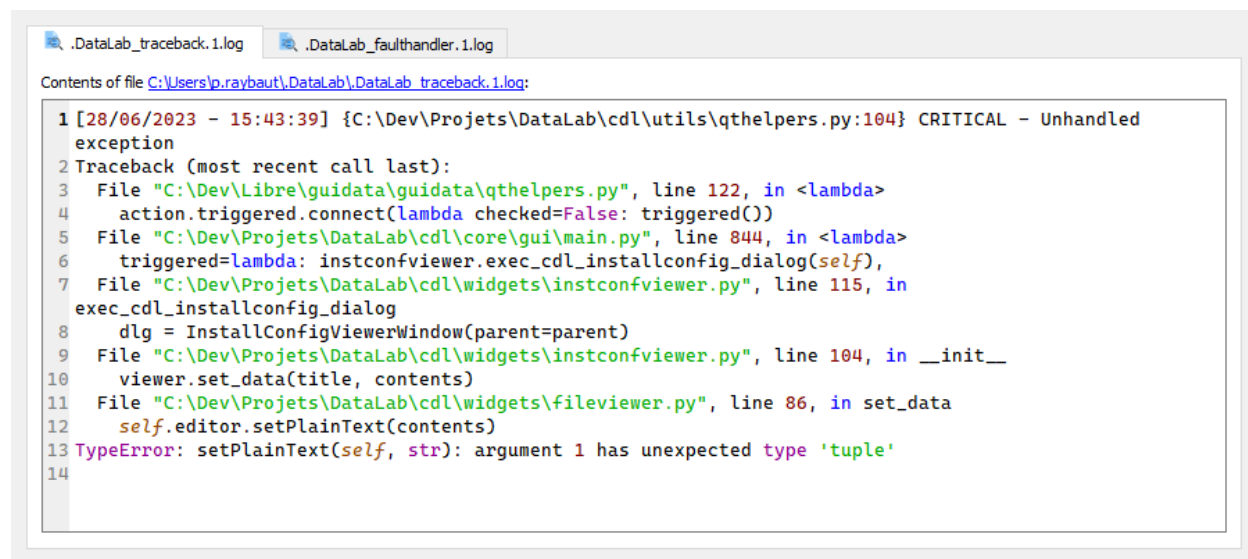
- « Traceback log », pour les exceptions Python
- « Faulthandler log », pour les erreurs système (p.ex. crash lié à Qt)

---

**Note :** The log viewer is accessible from the « ? » menu in the main window.

---

Lorsque DataLab s'arrête brutalement en raison d'une erreur système ou si une exception Python est levée durant son exécution, ces journaux de bord sont mis à jour en conséquence. DataLab notifie même l'utilisateur de la disponibilité de nouvelles informations dans les journaux de bord, lors du prochain démarrage de l'application. Ceci est une invitation à soumettre un rapport d'anomalie.



```

1 [28/06/2023 - 15:43:39] {C:\Dev\Projets\DataLab\cdl\utils\qthelpers.py:104} CRITICAL - Unhandled
exception
2 Traceback (most recent call last):
3   File "C:\Dev\Libre\guidata\guidata\qthelpers.py", line 122, in <lambda>
4     action.triggered.connect(lambda checked=False: triggered())
5   File "C:\Dev\Projets\DataLab\cdl\core\gui\main.py", line 844, in <lambda>
6     triggered=lambda: instconfviewer.exec_cdl_installconfig_dialog(self),
7   File "C:\Dev\Projets\DataLab\cdl\widgets\instconfviewer.py", line 115, in
exec_cdl_installconfig_dialog
8     dlg = InstallConfigViewerWindow(parent=parent)
9   File "C:\Dev\Projets\DataLab\cdl\widgets\instconfviewer.py", line 104, in __init__
10     viewer.set_data(title, contents)
11   File "C:\Dev\Projets\DataLab\cdl\widgets\fileviewer.py", line 86, in set_data
12     self.editor.setPlainText(contents)
13 TypeError: setPlainText(self, str): argument 1 has unexpected type 'tuple'
14

```

FIG. 6 – Journaux de bord DataLab (voir le menu « ? »)

Signaler un comportement inattendu ou tout autre type d'anomalie sur la page [GitHub Issues](#) sera grandement apprécié, d'autant plus si vous prenez soin de joindre le contenu des journaux de bord (ainsi que des informations sur votre configuration, cf. *Installation et configuration*).

## 2.2.8 Installation et configuration

En raison des multiples façons d'installer DataLab sur votre machine, comprendre l'origine d'un dysfonctionnement de l'application sans information sur votre configuration peut s'avérer très difficile.

C'est pourquoi DataLab fournit la boîte de dialogue « Installation et configuration » qui rassemble toutes les informations sur votre installation et votre configuration utilisateur.

---

**Note :** Le visualiseur d'installation et de configuration est accessible depuis le menu « ? » de la fenêtre principale.

---

Signaler un comportement inattendu ou tout autre type d'anomalie sur la page [GitHub Issues](#) sera grandement apprécié, d'autant plus si vous prenez soin de joindre les informations ci-dessus (ainsi que les journaux d'historique, cf. *Journaux de bord*).

## 2.2.9 Préférences

## 2.2.10 Ligne de commande

### Exécuter DataLab

Pour exécuter DataLab depuis la ligne de commande, taper la commande suivante :

```
$ datalab
```

Pour afficher l'aide de l'utilisation en ligne de commande, taper simplement :

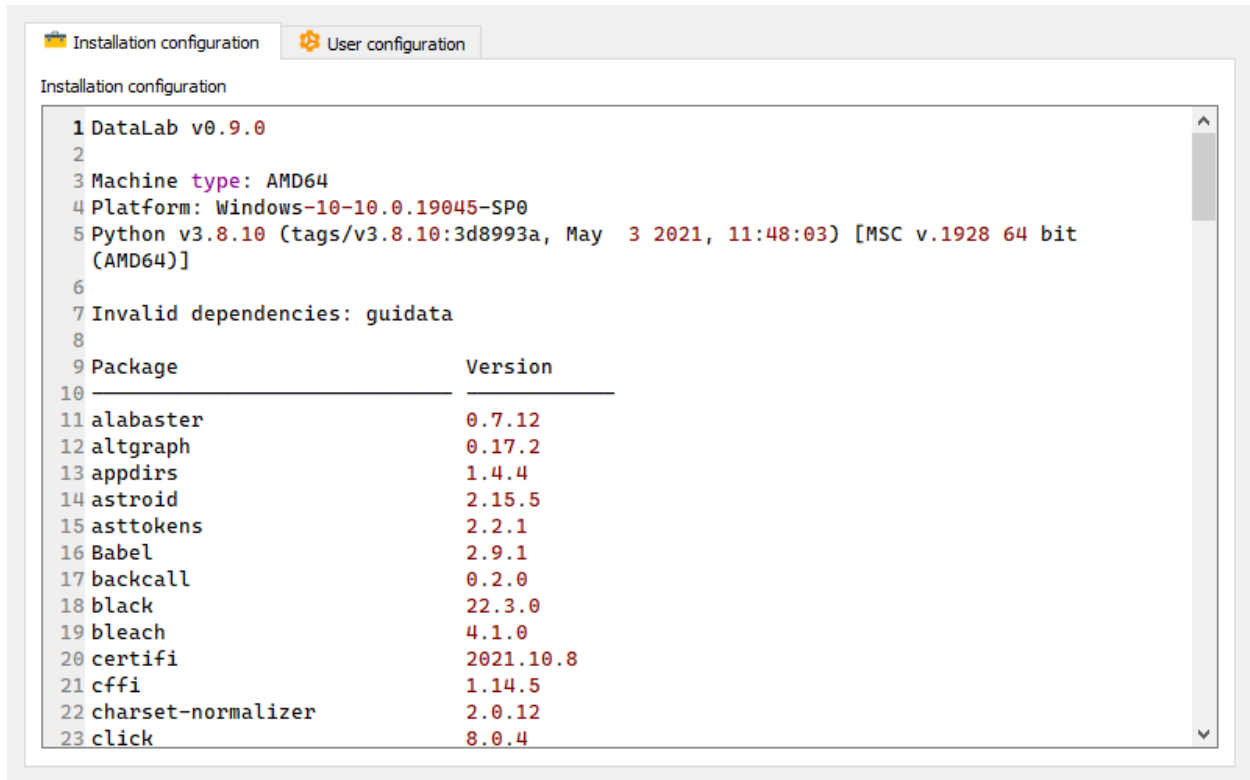


FIG. 7 – Installation et configuration (voir menu « ? »)

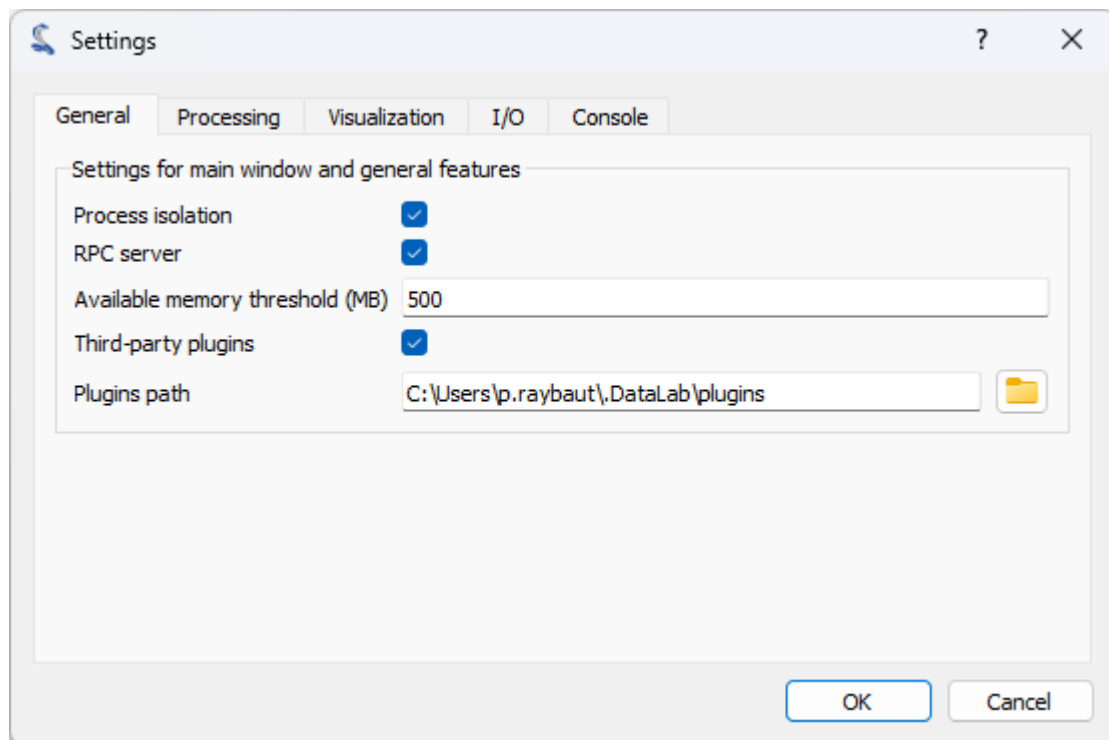


FIG. 8 – Capture d'écran de la boîte de dialogue « Préférences ».

```
$ datalab --help
usage: app.py [-h] [-b path] [-v] [--unattended] [--screenshot] [--delay DELAY] [--
↪xmlrpcport PORT]
               [--verbose {quiet,minimal,normal}]
               [h5]

Run DataLab

positional arguments:
h5                      HDF5 file names (separated by ';'), optionally with dataset name_
↪(separated by ',')
```

options:

-h, --help	show this help message and exit
-b path, --h5browser path	path to open with HDF5 browser
-v, --version	show DataLab version
--reset	reset DataLab configuration
--unattended	non-interactive mode
--accept_dialogs	accept dialogs in unattended mode
--screenshot	automatic screenshots
--delay DELAY	delay (ms) before quitting application in unattended mode
--xmlrpcport XMLRPCPORT	XML-RPC port number
--verbose {quiet,normal,debug}	verbosity level: for debugging/testing purpose

## Ouvrir un fichier HDF5 au démarrage

Pour ouvrir des fichiers HDF5, en important éventuellement un dataset précis du fichier HDF5, utiliser l'une des commandes suivantes :

```
$ datalab /path/to/file1.h5
$ datalab /path/to/file1.h5,/path/to/dataset1
$ datalab /path/to/file1.h5,/path/to/dataset1;/path/to/file2.h5,/path/to/dataset2
```

## Ouvrir l'explorateur de fichiers HDF5 au démarrage

Pour ouvrir l'explorateur de fichiers HDF5 au démarrage, utiliser l'une des commandes suivantes :

```
$ datalab -b /path/to/file1.h5
$ datalab --h5browser /path/to/file1.h5
```

## Mode démonstration de DataLab

Pour exécuter le mode de démonstration de DataLab, taper la commande suivante :

```
$ datalab-demo
```

## Exécuter les tests de validation technique

**Note :** Les tests de validation technique sont directement inclus dans les tests unitaires individuels et sont disséminés dans tout le code. Les fonctions de test incluant des tests de validation sont marquées avec le décorateur `@pytest.mark.validation`.

Pour exécuter les tests de validation technique de DataLab, taper la commande suivante :

```
$ pytest -m validation
```

### Voir aussi :

Voir la section [Validation](#) pour plus d'informations sur la stratégie de validation de DataLab.

## Exécuter l'ensemble des tests

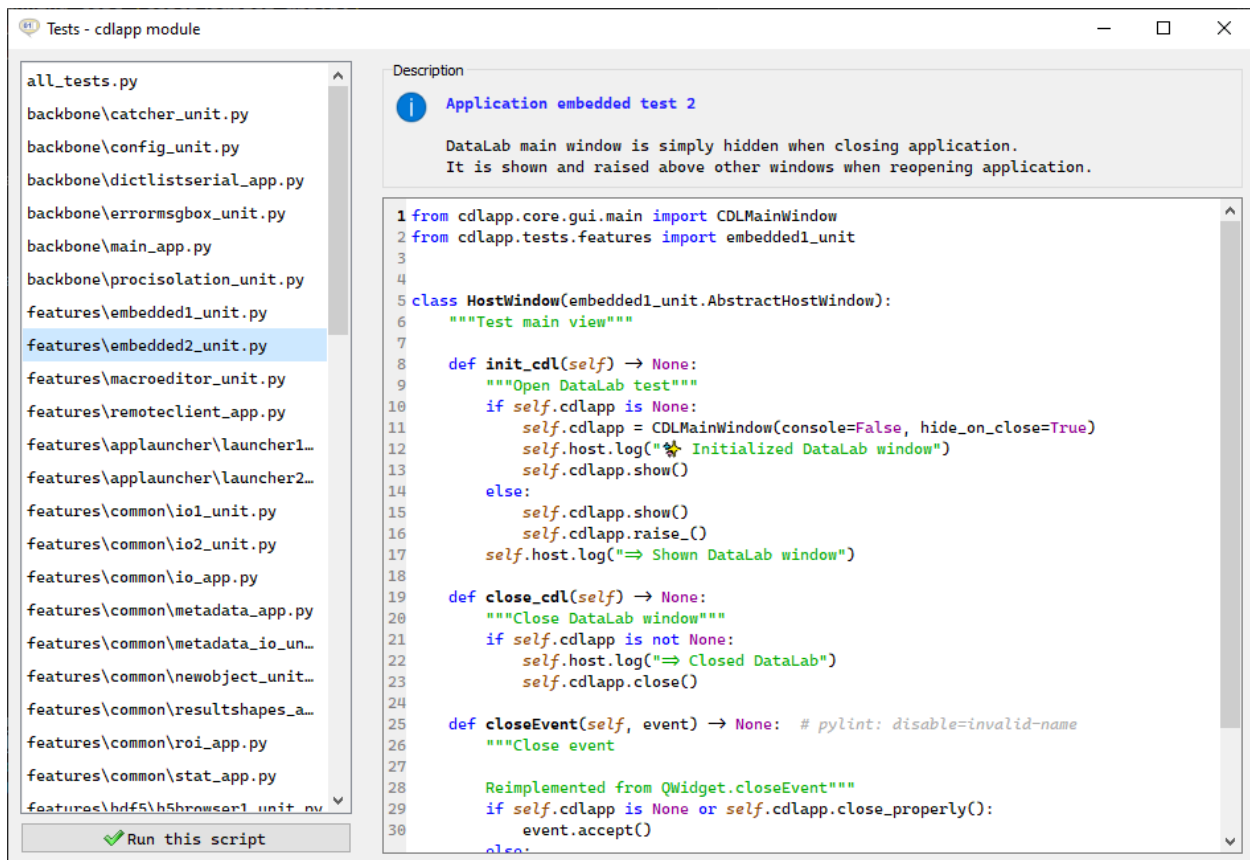
Pour exécuter l'ensemble des tests unitaires de DataLab, taper la commande suivante :

```
$ pytest
```

## Exécuter les tests interactifs

Pour exécuter les tests interactifs de DataLab, taper la commande suivante :

```
$ datalab-tests
```



## 2.3 Traitement du signal

Cette section décrit les fonctionnalités spécifiques au panneau de traitement du signal. Le panneau de traitement du signal est le panneau par défaut lorsque DataLab est démarré.

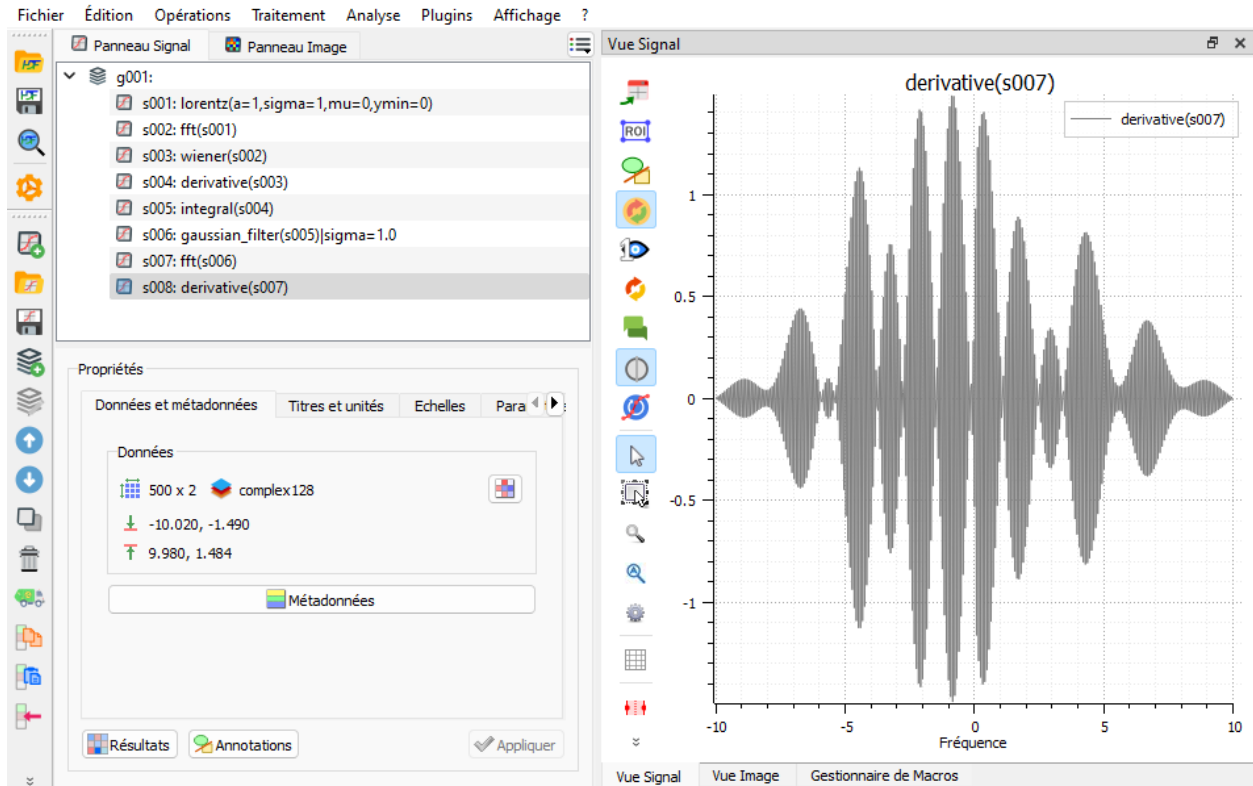


FIG. 9 – Fenêtre principale de DataLab : vue du traitement du signal

### 2.3.1 Créer, ouvrir et enregistrer des signaux

Cette section décrit comment créer, ouvrir et enregistrer des signaux (et des espaces de travail).

Lorsque le « Panneau Signal » est sélectionné, les menus et barres d'outils sont mis à jour pour fournir les actions liées aux signaux.

Le menu « Fichier » vous permet de :

- Créer, ouvrir, enregistrer et fermer des signaux (voir ci-dessous).
- Sauvegarder et restaurer l'espace de travail actuel ou parcourir les fichiers HDF5 (voir [Espace de travail](#)).
- Modifier les préférences de DataLab (voir [Préférences](#)).

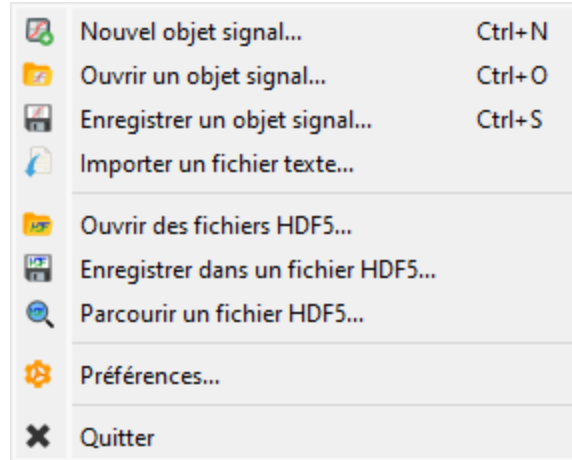


FIG. 10 – Capture d'écran du menu « Fichier ».

## Nouveau signal

Crée un nouveau signal depuis différents modèles :

Modèle	Equation
Zéros	$y[i] = 0$
Gaussienne	$y = y_0 + \frac{A}{\sqrt{2\pi} \cdot \sigma} \cdot \exp\left(-\frac{1}{2} \cdot \left(\frac{x - x_0}{\sigma}\right)^2\right)$
Lorentzienne	$y = y_0 + \frac{A}{\sigma \cdot \pi} \cdot \frac{1}{1 + \left(\frac{x - x_0}{\sigma}\right)^2}$
Voigt	$y = y_0 + A \cdot \frac{\text{Re}(\exp(-z^2)) \cdot \text{erfc}(-j \cdot z)}{\sqrt{2\pi} \cdot \sigma}$ avec $z = \frac{x - x_0 - j \cdot \sigma}{\sqrt{2} \cdot \sigma}$
Aléatoire (loi uniforme)	$y[i] \in [-0.5, 0.5]$
Aléatoire (loi normale)	$y[i] \sim \mathcal{N}(-0.5, 0.5)$
Sinus	$y = y_0 + A \cdot \sin(2\pi \cdot f \cdot x + \phi)$
Cosinus	$y = y_0 + A \cdot \cos(2\pi \cdot f \cdot x + \phi)$
Dents de scie	$y = y_0 + A \cdot \left(2 \left(fx + \frac{\phi}{2\pi} - \left\lfloor fx + \frac{\phi}{2\pi} + \frac{1}{2} \right\rfloor\right)\right)$
Triangle	$y = y_0 + A \cdot \text{sawtooth}(2\pi f x + \phi, \text{width} = 0.5)$
Carré	$y = y_0 + A \cdot \text{sgn}(\sin(2\pi f x + \phi))$
Sinus cardinal	$y = y_0 + A \cdot \text{sinc}(2\pi f x + \phi)$
Echelon	$y = y_0 + A \cdot \begin{cases} 1 & \text{if } x > x_0 \\ 0 & \text{otherwise} \end{cases}$
Exponentielle	$y = y_0 + A \cdot \exp(B \cdot x)$
Impulsion	$y = y_0 + A \cdot \begin{cases} 1 & \text{if } x_0 < x < x_1 \\ 0 & \text{otherwise} \end{cases}$
Polynomial	$y = y_0 + A_0 + A_1 \cdot x + A_2 \cdot x^2 + \dots + A_n \cdot x^n$
Expérimental	Saisie manuelle des valeurs X et Y

## Ouvrir un signal

Crée un signal depuis l'un des types de fichiers pris en charge :

Type de fichier	Extensions
Fichiers texte	.txt, .csv
Tableaux NumPy	.npy
Fichiers MAT	.mat

## Enregistrer un signal

Enregistre le signal sélectionné dans l'un des types de fichier pris en charge :

Type de fichier	Extensions
Fichiers texte	.csv

## Importer un fichier texte

DataLab peut importer nativement des fichiers de signaux (par exemple CSV, NPY, etc.). Cependant, certains formats de fichiers texte spécifiques peuvent ne pas être pris en charge. Dans ce cas, vous pouvez utiliser la fonctionnalité « Importer un fichier texte », qui vous permet d'importer un fichier texte et de le convertir en signal.

Cette fonctionnalité est accessible depuis le menu « Fichier », sous l'option « Importer un fichier texte ».

Il ouvre un assistant d'importation qui vous guide tout au long du processus d'importation du fichier texte.

## Etape 1 : Sélectionner la source

La première étape consiste à sélectionner la source du fichier texte. Vous pouvez soit sélectionner un fichier de votre ordinateur, soit le presse-papiers si vous avez copié le texte à partir d'une autre application.

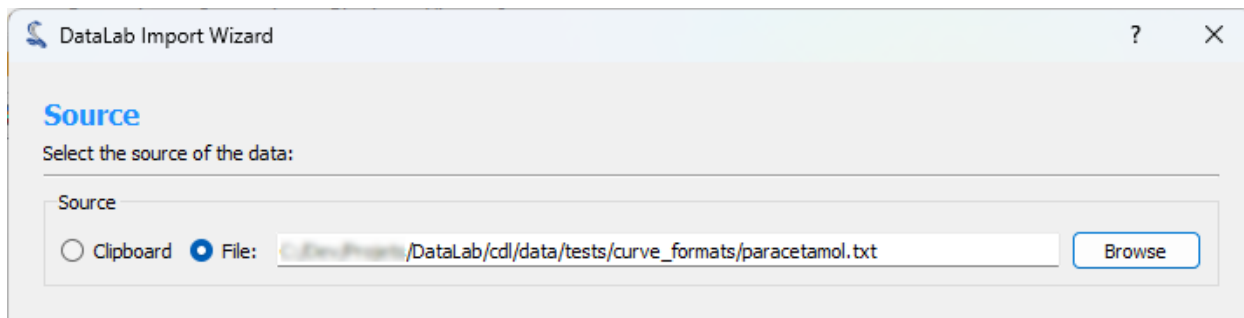


FIG. 11 – Etape 1 : Sélectionner la source

## Etape 2 : Aperçu et configuration de l'importation

La deuxième étape consiste à configurer l'importation et à prévisualiser le résultat. Vous pouvez configurer les options suivantes :

- **Délimiteur** : Le caractère utilisé pour séparer les valeurs dans le fichier texte.
- **Commentaires** : Le caractère utilisé pour indiquer que la ligne est un commentaire et doit être ignorée.
- **Lignes à sauter** : Le nombre de lignes à sauter au début du fichier.
- **Nombre maximum de lignes** : Le nombre maximum de lignes à importer. Si le fichier contient plus de lignes, elles seront ignorées.
- **Transposer** : Si coché, les lignes et les colonnes seront transposées.
- **Type de données** : Le type de données de destination des données importées.
- **La première colonne est X** : Si coché, la première colonne sera utilisée comme axe X.

Lorsque vous avez terminé de configurer l'importation, cliquez sur le bouton « Appliquer » pour voir le résultat.

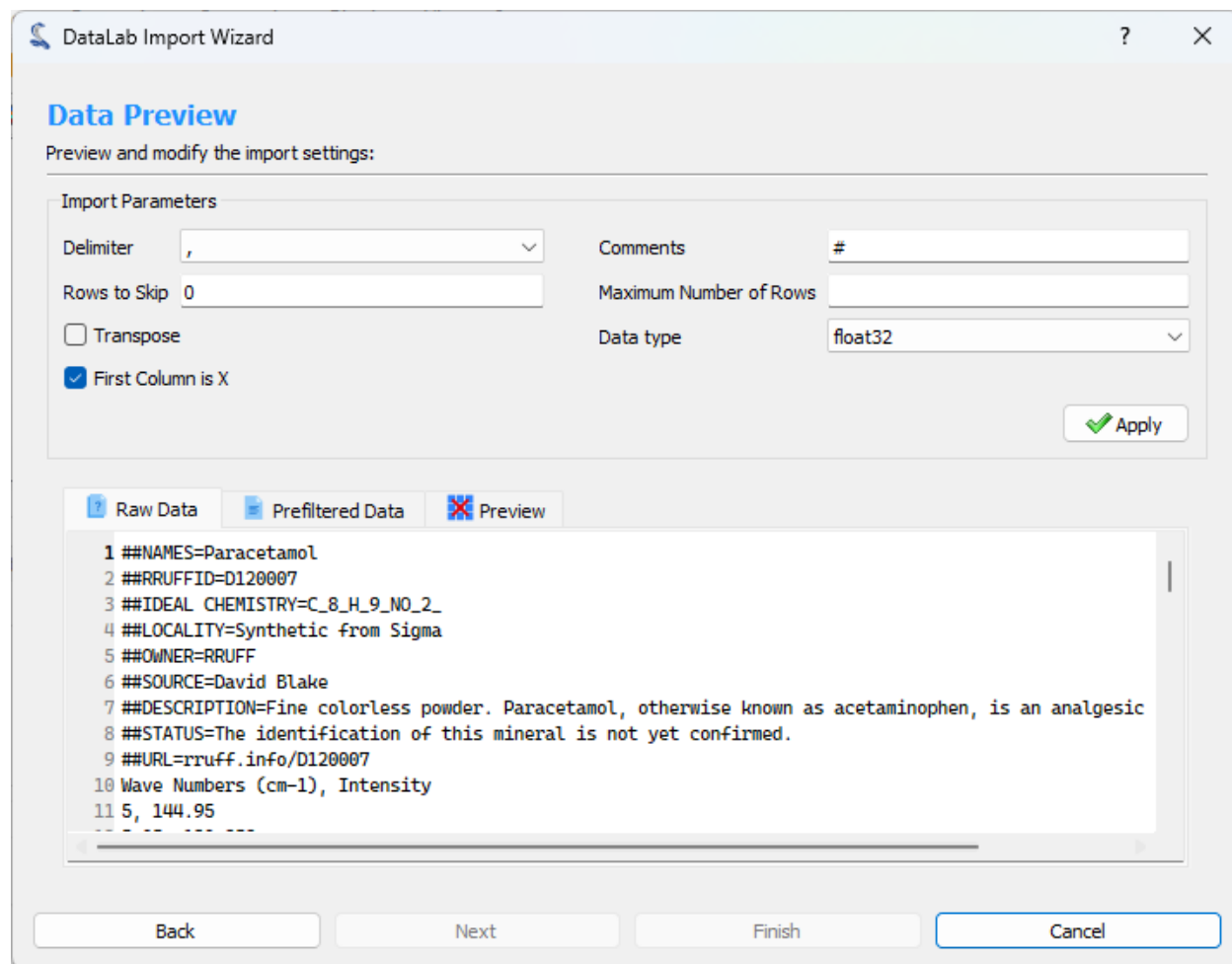


FIG. 12 – Etape 2 : Configurer l'importation

DataLab Import Wizard

### Data Preview

Preview and modify the import settings:

Import Parameters

Delimiter: ,

Comments: #

Rows to Skip: 10

Maximum Number of Rows:

☐ Transpose

☒ First Column is X

Data type: float32

Apply

Raw Data Prefiltered Data Preview

	X	Y
1	5.0	144.95
2	5.05	130.958
3	5.1	138.617
4	5.15	127.86
5	5.2	130.526
6	5.25	129.117

Back Next Finish Cancel

FIG. 13 – Etape 2 : Prévisualiser le résultat

### Etape 3 : Afficher la représentation graphique

La troisième étape montre une représentation graphique des données importées. Vous pouvez utiliser le bouton « Terminer » pour importer les données dans l'espace de travail de DataLab.

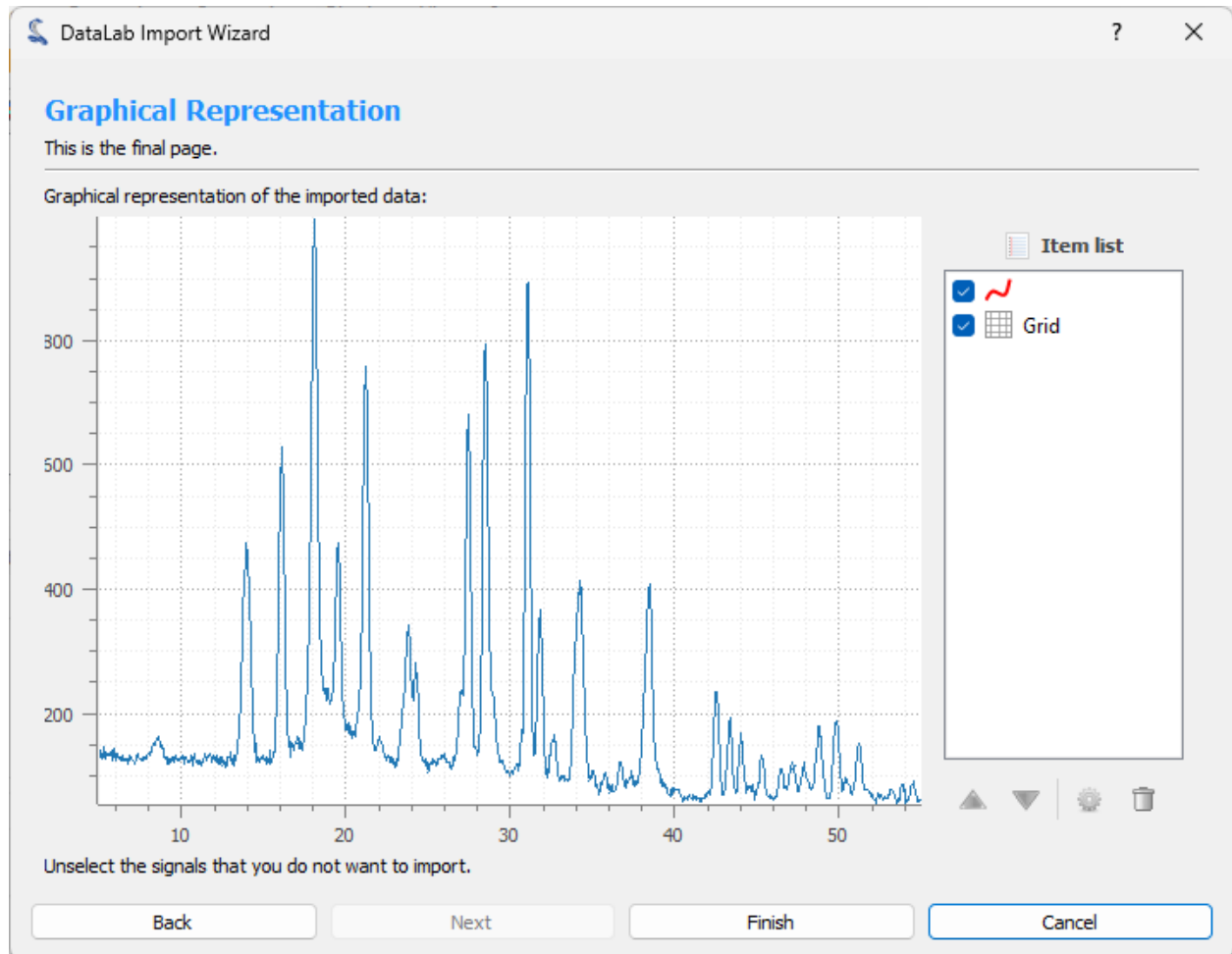


FIG. 14 – Etape 3 : Afficher la représentation graphique

### 2.3.2 Manipuler les métadonnées

Cette section décrit comment manipuler les métadonnées dans DataLab.

Le menu « Édition » vous permet d'effectuer des opérations d'édition classiques sur le signal ou le groupe de signaux actuel (créer/renommer un groupe, déplacer vers le haut/vers le bas, supprimer le signal/le groupe de signaux, etc.).

Il vous permet également de manipuler les métadonnées associées au signal actuel.

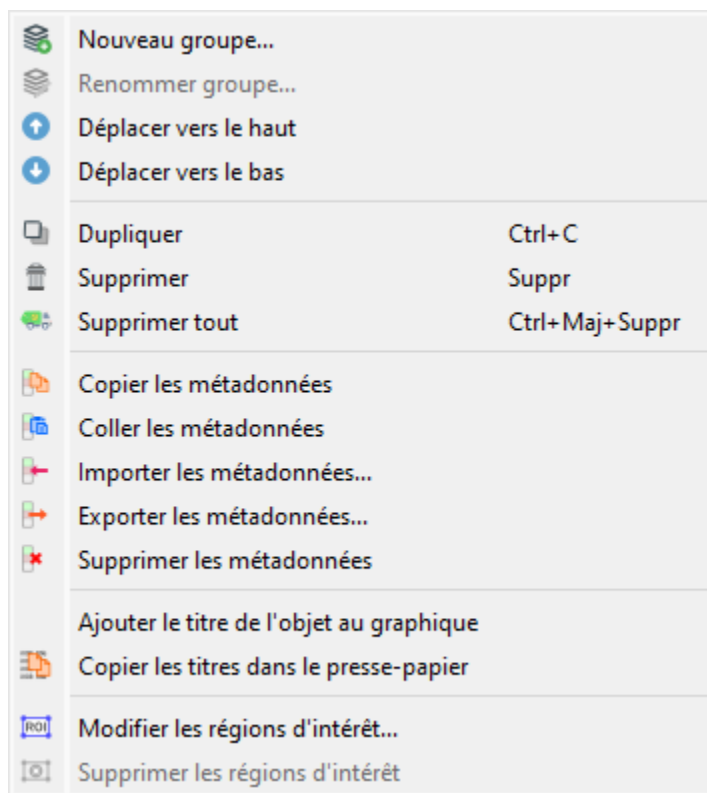


FIG. 15 – Capture d'écran du menu « Édition ».

### Copier/coller les métadonnées

Compte tenu du fait que les métadonnées contiennent des informations utiles sur le signal, elles peuvent être copiées et collées d'un signal à un autre en sélectionnant les actions « Copier les métadonnées » et « Coller les métadonnées » dans le menu « Édition ».

Cette fonctionnalité vous permet de transférer ces informations d'un signal à un autre :

- *Régions d'intérêt (ROIs)* : c'est un moyen très efficace de réutiliser la même ROI sur différents signaux et de comparer facilement les résultats de l'analyse sur ces signaux
- Résultats de calcul, tels que les positions de pics ou les intervalles FWHM (la pertinence du transfert de ces informations dépend du contexte et revient à l'utilisateur de décider)
- Toute autre information que vous avez pu ajouter aux métadonnées d'un signal

---

**Note :** Copier les métadonnées d'un signal à un autre écrasera les métadonnées du signal de destination (pour les clés de métadonnées communes aux deux signaux) ou ajoutera simplement les clés de métadonnées qui ne sont pas présentes dans le signal de destination.

---

## Importer/exporter les métadonnées

Les métadonnées peuvent également être importées et exportées depuis/vers un fichier JSON en utilisant les actions « Importer les métadonnées » et « Exporter les métadonnées » dans le menu « Édition ». C'est exactement la même chose que la fonctionnalité de copier/coller des métadonnées (voir ci-dessus pour plus de détails sur les cas d'utilisation de cette fonctionnalité), mais cela vous permet de sauvegarder les métadonnées dans un fichier et de les importer ultérieurement.

## Supprimer les métadonnées

Lors de la suppression des métadonnées en utilisant l'action « Supprimer les métadonnées » dans le menu « Édition », vous serez invité à confirmer la suppression des régions d'intérêt (ROIs) si elles sont présentes dans les métadonnées. Après cette confirmation éventuelle, les métadonnées seront supprimées, ce qui signifie que les résultats d'analyse, les ROIs et toute autre information associée au signal seront perdus.

## Titres des signaux

Les titres des signaux peuvent être considérés comme des métadonnées du point de vue de l'utilisateur, même s'ils ne sont pas stockés dans les métadonnées du signal (mais dans un attribut de l'objet signal).

Le menu « Édition » vous permet de :

- « Ajouter le titre de l'objet au graphique » : cette action ajoutera une étiquette en haut du signal avec son titre.
- « Copier les titres dans le presse-papiers » : cette action copiera les titres des signaux sélectionnés dans le presse-papiers, ce qui peut être utile pour les coller dans un éditeur de texte ou dans un tableur.

Exemple du contenu du presse-papiers :

```
g001:
  s001: lorentz(a=1,sigma=1,mu=0,ymin=0)
  s002: derivative(s001)
  s003: wiener(s002)
g002: derivative(g001)
  s004: derivative(s001)
  s005: derivative(s002)
  s006: derivative(s003)
g003: fft(g002)
  s007: fft(s004)
  s008: fft(s005)
  s009: fft(s006)
```

## Régions d'intérêt (ROI)

Les régions d'intérêt (ROI) sont des zones de signal définies par l'utilisateur pour effectuer des opérations, des traitements ou des analyses spécifiques sur elles.

Les ROI sont prises en compte presque partout dans les fonctionnalités de calcul de DataLab :

- Les fonctionnalités du menu « Opérations » sont effectuées uniquement sur la ROI si elle est définie (sauf si l'opération modifie le nombre de points, comme l'interpolation ou le rééchantillonnage).
- Les actions du menu « Traitement » sont effectuées uniquement sur la ROI si elle est définie (sauf si le type de données du signal de destination est différent de celui de la source, comme dans les fonctionnalités d'analyse de Fourier).
- Les actions du menu « Analyse » sont effectuées uniquement sur la ROI si elle est définie.

**Note :** Les ROI sont stockées en tant que métadonnées et sont donc attachées au signal.

Le menu « Édition » vous permet de :

- « Modifier les régions d'intérêt » : ouvre une boîte de dialogue pour gérer les ROI associées au signal sélectionné (ajouter, supprimer, déplacer, redimensionner, etc.). La boîte de dialogue de définition des ROI est exactement la même que l'extraction des ROI (voir ci-dessous) : le ROI est défini en déplaçant la position et en ajustant la largeur d'une plage horizontale.

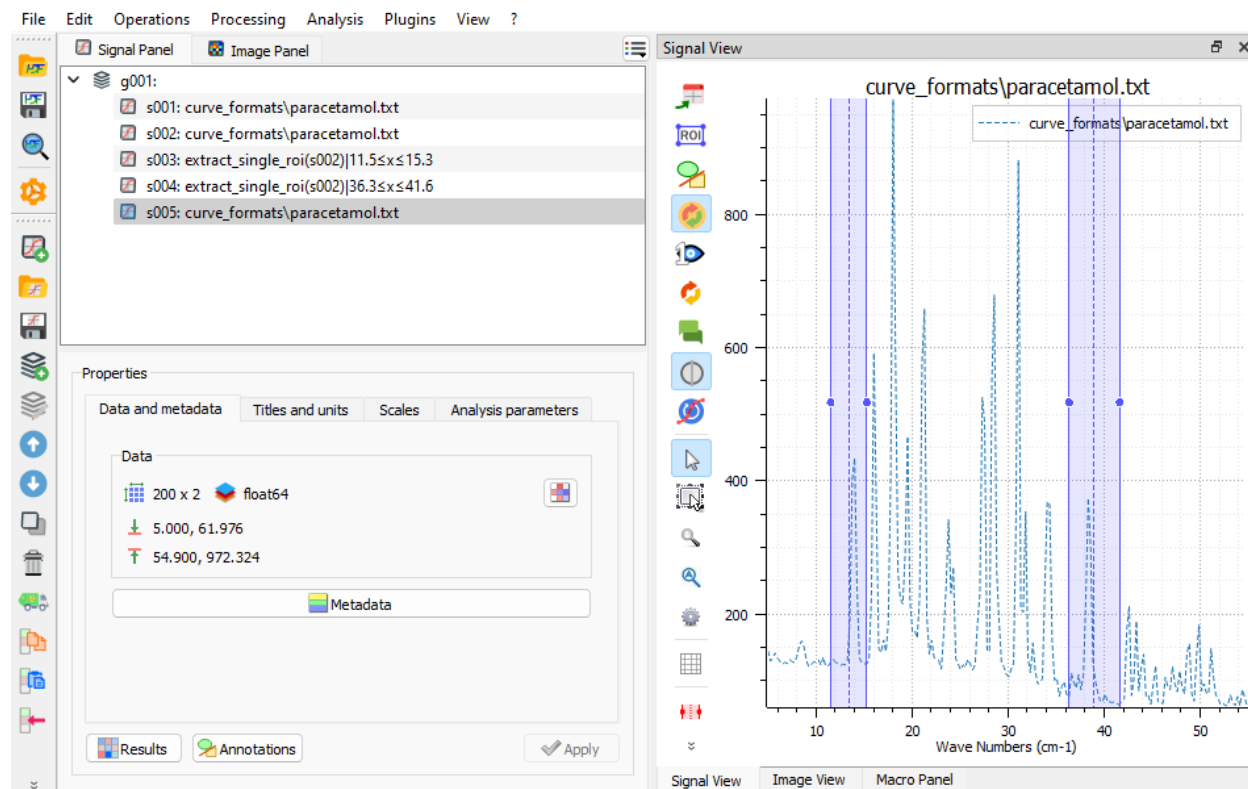


FIG. 16 – Un signal avec une ROI.

- « Supprimer les régions d'intérêt » : supprime toutes les ROI définies pour les signaux sélectionnés.

### 2.3.3 Opérations sur les signaux

Cette section décrit les opérations qui peuvent être effectuées sur les signaux.

**Voir aussi :**

*Traitement des signaux* pour plus d'informations sur les fonctionnalités de traitement des signaux, ou *Analyse sur les signaux* pour des informations sur les fonctionnalités d'analyse des signaux.

Lorsque le « Panneau Signal » est sélectionné, les menus et barres d'outils sont mis à jour pour fournir les actions liées aux signaux.

Le menu « Opérations » permet d'effectuer diverses opérations sur les signaux sélectionnés, telles que des opérations arithmétiques, la détection de pics, ou encore la convolution.

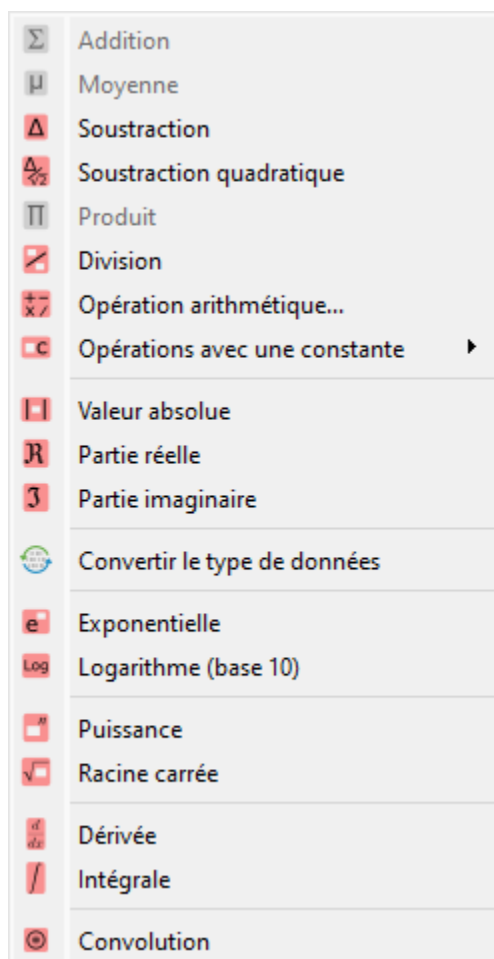


FIG. 17 – Capture d'écran du menu « Opérations ».

## Opérations arithmétiques de base

Opération	Description
Somme	$y_M = \sum_{k=0}^{M-1} y_k$
Moyenne	$y_M = \frac{1}{M} \sum_{k=0}^{M-1} y_k$
Différence	$y_2 = y_1 - y_0$
Produit	$y_M = \prod_{k=0}^{M-1} y_k$
Division	$y_2 = \frac{y_1}{y_0}$

## Opérations avec une constante

Crée un signal à partir d'une opération avec une constante sur chaque signal sélectionné :

Opération	Description
Addition	$y_k = y_{k-1} + c$
Soustraction	$y_k = y_{k-1} - c$
Multiplication	$y_k = y_{k-1} \times c$
Division	$y_k = \frac{y_{k-1}}{c}$

## Valeur absolue, parties réelle et imaginaire

Opération	Description
Valeur absolue	$y_k =  y_{k-1} $
Partie réelle	$y_k = \Re(y_{k-1})$
Partie imaginaire	$y_k = \Im(y_{k-1})$

## Conversion du type de données

L'action « Convertir le type de données » permet de convertir le type de données des signaux sélectionnés.

**Note :** La conversion du type de données utilise la fonction `numpy.ndarray.astype()` avec les paramètres par défaut (`casting="unsafe"`).

## Fonctions mathématiques de base

Fonction	Description
Exponentielle	$y_k = \exp(y_{k-1})$
Logarithme (base 10)	$y_k = \log_{10}(y_{k-1})$
Puissance	$y_k = y_{k-1}^n$
Racine carrée	$y_k = \sqrt{y_{k-1}}$

## Autres opérations mathématiques

Opération	Implémentation
Dérivée	Basée sur <code>numpy.gradient</code>
Intégrale	Basée sur <code>scipy.integrate.cumulative_trapezoid</code>
Convolution	Basée sur <code>scipy.signal.convolve</code>

## 2.3.4 Traitement des signaux

Cette section décrit les fonctionnalités de traitement de signal disponibles dans DataLab.

### Voir aussi :

*Opérations sur les signaux* pour plus d'informations sur les opérations qui peuvent être effectuées sur les signaux, ou *Analyse sur les signaux* pour des informations sur les fonctionnalités d'analyse des signaux.

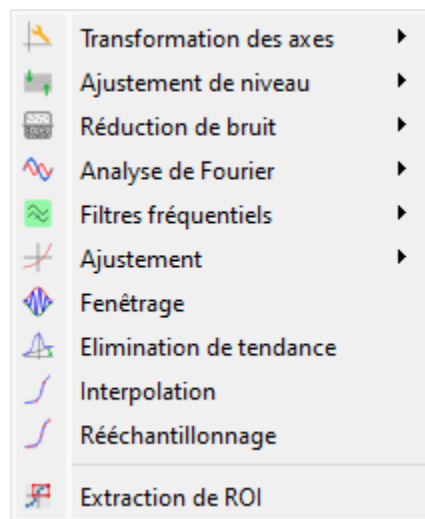


FIG. 18 – Capture d'écran du menu « Traitement ».

Lorsque le « Panneau Signal » est sélectionné, les menus et barres d'outils sont mis à jour pour fournir les actions liées aux signaux.

Le menu « Traitement » permet d'effectuer divers traitements sur les signaux sélectionnés, tels que le lissage, la normalisation, ou encore l'interpolation.

## Transformation des axes

### Étalonnage linéaire

Crée un signal à partir de l'étalonnage linéaire (par rapport aux axes X et Y) de chaque signal sélectionné.

Paramètre	Étalonnage linéaire
Axe des X	$x_1 = a.x_0 + b$
Axe des Y	$y_1 = a.y_0 + b$

### Permuter les axes X/Y

Crée un signal à partir des données inversées X/Y du signal sélectionné.

### Inverser l'axe des X

Crée un signal à partir des données inversées de l'axe des X du signal sélectionné.

## Ajustement des niveaux

### Normaliser

Crée un signal à partir de la normalisation de chaque signal sélectionné par maximum, amplitude, somme, énergie ou RMS :

Paramètre	Normalisation
Maximum	$y_1 = \frac{y_0}{\max(y_0)}$
Amplitude	$y_1 = \frac{y'_0}{\max(y'_0)}$ avec $y'_0 = y_0 - \min(y_0)$
Aire	$y_1 = \frac{\sum_{n=0}^N y_0[n]}{y_0}$
Energie	$y_1 = \frac{\sqrt{\sum_{n=0}^N  y_0[n] ^2}}{y_0}$
RMS	$y_1 = \frac{\sqrt{\frac{1}{N} \sum_{n=0}^N  y_0[n] ^2}}{y_0}$

## Ecrêtage

Crée un signal à partir de l'écrtage de chaque signal sélectionné.

## Soustraction d'offset

Crée un signal à partir de la soustraction d'offset de chaque signal sélectionné. Cette opération est réalisée en soustrayant la valeur moyenne d'une plage définie par l'utilisateur.

## Réduction de bruit

Crée un signal à partir du débruitage de chaque signal sélectionné.

Les filtres suivants sont disponibles :

Filtre	Formule/implémentation
Filtre gaussien	<code>scipy.ndimage.gaussian_filter</code>
Moyenne mobile	<code>scipy.ndimage.uniform_filter</code>
Médiane mobile	<code>scipy.ndimage.median_filter</code>
Filtre de Wiener	<code>scipy.signal.wiener</code>

## Analyse de Fourier

Crée un signal à partir de l'analyse de Fourier de chaque signal sélectionné.

Les fonctions suivantes sont disponibles :

Fonction	Description	Formule/implémentation
FFT	Transformée de Fourier rapide	<code>numpy.fft.fft</code>
FFT inverse	Transformée de Fourier rapide inverse	<code>numpy.fft.ifft</code>
Spectre d'amplitude	Optionnel : utiliser une échelle logarithmique (dB)	$y_1 =  FFT(y_0) $ ou $20 \cdot \log_{10}( FFT(y_0) )$ (dB)
Spectre de phase		$y_1 = \angle FFT(y_0)$
Densité spectrale de puissance (PSD)	Optionnel : utiliser une échelle logarithmique (dB). La PSD est estimée en utilisant la méthode de Welch (voir <code>scipy.signal.welch</code> )	$Y_k = PSD(y_k)$ ou $10 \cdot \log_{10}(PSD(y_k))$ (dB)

**Note :** La FFT et la FFT inverse sont effectuées en utilisant un décalage de fréquence si l'option est activée dans les paramètres de DataLab (voir [Préférences](#)).

## Filtres fréquentiels

Crée un signal à partir de l'application d'un filtre fréquentiel à chaque signal sélectionné.

Les filtres suivants sont disponibles :

Filtre	Description
Low-pass	Filtre les hautes fréquences, au-dessus d'une fréquence de coupure
High-pass	Filtre les basses fréquences, en-dessous d'une fréquence de coupure
Band-pass	Filtre les fréquences en dehors d'une plage
Band-stop	Filtre les fréquences à l'intérieur d'une plage

Pour chaque filtre, les méthodes suivantes sont disponibles :

Méthode	Description
Bessel	Filtre de Bessel, utilisant la fonction <code>scipy.signal.bessel</code>
Butterworth	Filtre de Butterworth, utilisant la fonction <code>scipy.signal.butter</code>
Chebyshev I	Filtre de Chebyshev de type I, utilisant la fonction <code>scipy.signal.cheby1</code>
Chebyshev II	Filtre de Chebyshev de type II, utilisant la fonction <code>scipy.signal.cheby2</code>
Elliptic	Filtre elliptique, utilisant la fonction <code>scipy.signal.ellip</code>

## Ajustement

Ouvre une boîte de dialogue permettant de réaliser des ajustements de courbe de manière interactive.

Modèle	Equation
Linéaire	$y = c_0 + c_1.x$
Polynomial	$y = c_0 + c_1.x + c_2.x^2 + \dots + c_n.x^n$
Gaussienne	$y = y_0 + \frac{A}{\sqrt{2\pi}.\sigma} \cdot \exp\left(-\frac{1}{2} \cdot \left(\frac{x - x_0}{\sigma}\right)^2\right)$
Lorentzienne	$y = y_0 + \frac{A}{\sigma.\pi} \cdot \frac{1}{1 + \left(\frac{x - x_0}{\sigma}\right)^2}$
Voigt	$y = y_0 + A \cdot \frac{\text{Re}(\exp(-z^2)) \cdot \text{erfc}(-j.z)}{\sqrt{2\pi}.\sigma}$ avec $z = \frac{x - x_0 - j.\sigma}{\sqrt{2}.\sigma}$
Multi-gaussien	$y = y_0 + \sum_{i=0}^K \frac{A_i}{\sqrt{2\pi}.\sigma_i} \cdot \exp\left(-\frac{1}{2} \cdot \left(\frac{x - x_{0,i}}{\sigma_i}\right)^2\right)$
Exponentielle	$y = y_0 + A \cdot \exp(B.x)$
Sinusoidal	$y = y_0 + A \cdot \sin(2\pi.f.x + \phi)$
Fonction de distribution cumulative (CDF)	$y = y_0 + A \cdot \text{erf}\left(\frac{x - x_0}{\sigma.\sqrt{2}}\right)$

## Fenêtrage

Crée un signal à partir de l'application d'une fonction de fenêtrage à chaque signal sélectionné.

Les fonctions de fenêtrage suivantes sont disponibles :

Fonction de fenêtrage	Référence
Barthann	<code>scipy.signal.windows.barthann()</code>
Bartlett	<code>numpy.bartlett()</code>
Blackman	<code>scipy.signal.windows.blackman()</code>
Blackman-Harris	<code>scipy.signal.windows.blackmanharris()</code>
Bohman	<code>scipy.signal.windows.bohman()</code>
Boxcar	<code>scipy.signal.windows.boxcar()</code>
Cosinus	<code>scipy.signal.windows.cosine()</code>
Exponentielle	<code>scipy.signal.windows.exponential()</code>
Flat top	<code>scipy.signal.windows.flattop()</code>
Gaussienne	<code>scipy.signal.windows.gaussian()</code>
Hamming	<code>numpy.hamming()</code>
Hanning	<code>numpy.hanning()</code>
Kaiser	<code>scipy.signal.windows.kaiser()</code>
Lanczos	<code>scipy.signal.windows.lanczos()</code>
Nuttall	<code>scipy.signal.windows.nuttall()</code>
Parzen	<code>scipy.signal.windows.parzen()</code>
Rectangulaire	<code>numpy.ones()</code>
Taylor	<code>scipy.signal.windows.taylor()</code>
Tukey	<code>scipy.signal.windows.tukey()</code>

## Elimination de tendance

Crée un signal à partir de l'élimination de tendance de chaque signal. Cette fonctionnalité est basée sur la fonction `scipy.signal.detrend` de SciPy.

Les paramètres suivants sont disponibles :

Paramètre	Description
Méthode	Méthode d'élimination de tendance : "linéaire" ou "constante". Voir la fonction <code>scipy.signal.detrend</code> de SciPy.

## Interpolation

Crée un signal à partir de l'interpolation de chaque signal sélectionné, par rapport à l'axe X d'un second signal (qui peut être l'un des signaux sélectionnés).

Les méthodes d'interpolation suivantes sont disponibles :

Méthode	Description
Linéaire	Interpolation linéaire, utilisant la fonction <code>interp</code> de NumPy.
Spline	Interpolation par spline cubique, utilisant la fonction <code>scipy.interpolate.splev</code> de SciPy.
Quadratique	Interpolation quadratique, utilisant la fonction <code>polyval</code> de NumPy.
Cubique	Interpolation cubique, utilisant la classe <code>Akima1DInterpolator</code> de SciPy.
Barycentrique	Interpolation barycentrique, utilisant la classe <code>BarycentricInterpolator</code> de SciPy.
PCHIP	Interpolation par polynôme de Hermite cubique par morceaux (PCHIP), utilisant la classe <code>PchipInterpolator</code> de SciPy.

## Rééchantillonnage

Crée un signal à partir du rééchantillonnage de chaque signal.

Les paramètres suivants sont disponibles :

Paramètre	Description
Méthode	Méthode d'interpolation (voir section précédente)
Valeur de remplissage	Valeur de remplissage pour l'interpolation (voir section précédente)
Xmin	Borne inférieure
Xmax	Borne supérieure
Mode	Mode de rééchantillonnage : pas ou nombre de points
Pas	Pas de rééchantillonnage
Nombre de points	Nombre de points de rééchantillonnage

## Extraction de ROI

Crée un signal à partir d'une région d'intérêt (ROI) définie par l'utilisateur.

### 2.3.5 Analyse sur les signaux

Cette section décrit les fonctionnalités d'analyse de signaux disponibles dans DataLab.

#### Voir aussi :

*Opérations sur les signaux* pour plus d'informations sur les opérations qui peuvent être effectuées sur les signaux, ou *Traitement des signaux* pour des informations sur les fonctionnalités de traitement des signaux.

Lorsque le « Panneau Signal » est sélectionné, les menus et barres d'outils sont mis à jour pour fournir les actions liées aux signaux.

Le menu Analyse » permet d'effectuer divers calculs sur les signaux sélectionnés, tels que des statistiques, la largeur à mi-hauteur, ou encore la largeur à  $1/e^2$ .

**Note :** Dans le vocabulaire de DataLab, un « analyse » est une fonctionnalité qui calcule un résultat scalaire à partir d'un signal. Ce résultat est stocké sous la forme de métadonnées ; il est donc attaché au signal. Cela diffère d'un « traitement » qui crée un nouveau signal à partir d'un signal existant.

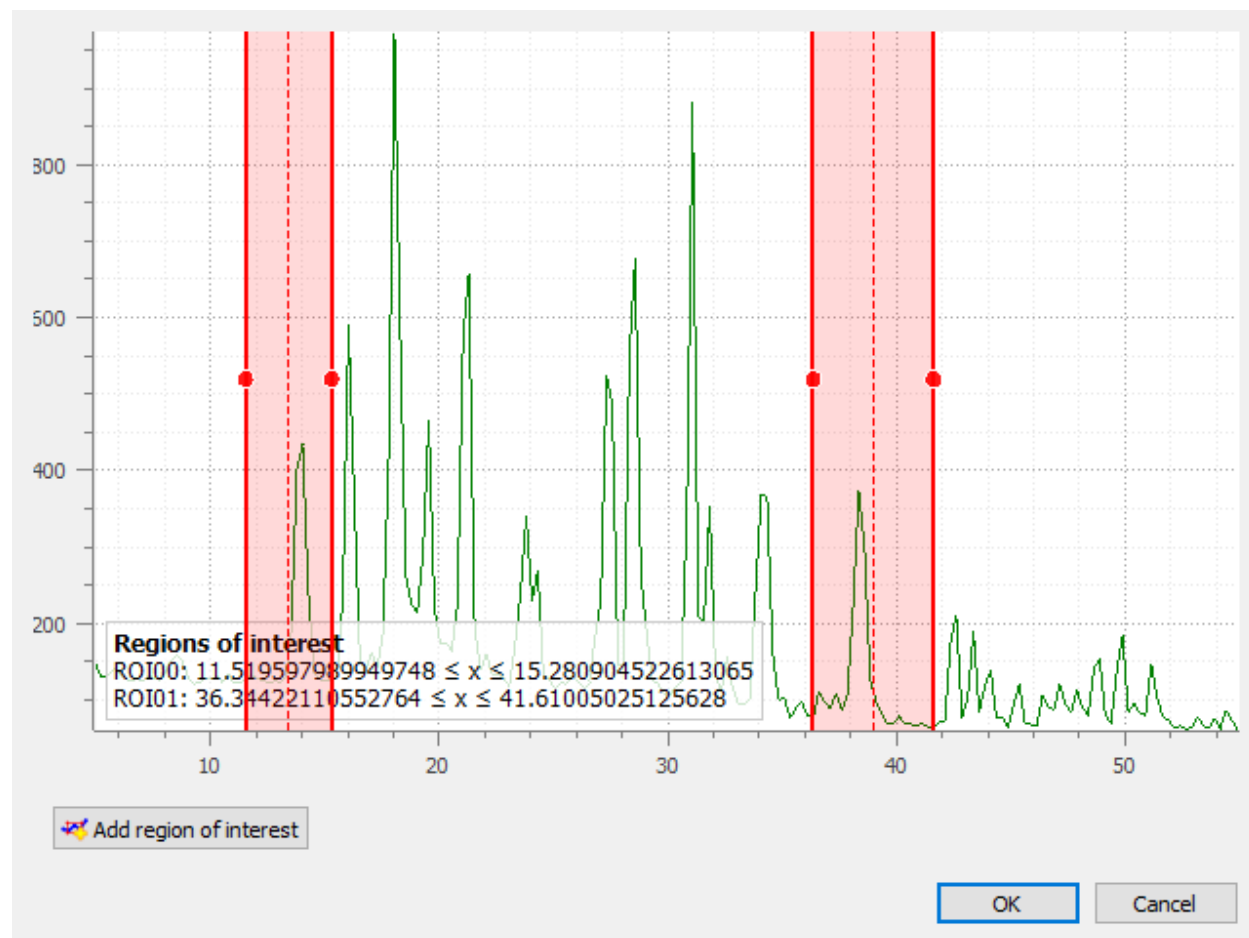


FIG. 19 – Boîte de dialogue d'extraction de ROI : la région d'intérêt (ROI) est définie en ajustant la position et la largeur de l'échelle horizontale de sélection.

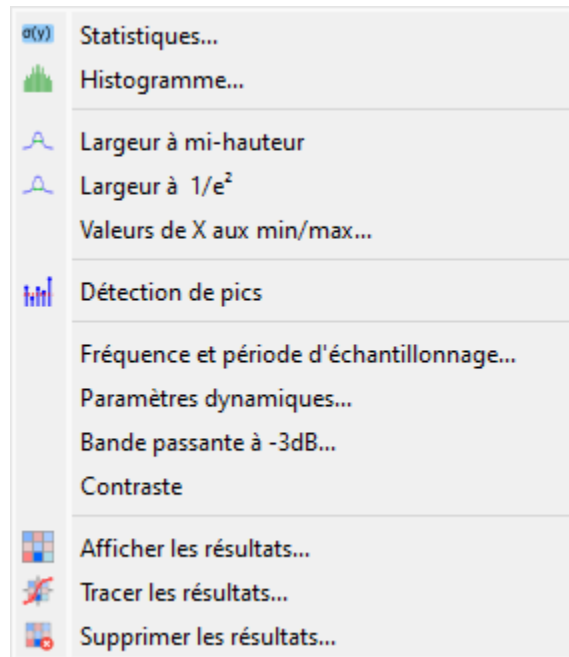


FIG. 20 – Capture d'écran du menu Analyse ».

### Statistiques

Calcule des statistiques sur les signaux sélectionnés et affiche un tableau récapitulatif.

### Histogramme

Calcule l'histogramme du signal sélectionné et l'affiche.

Paramètres :

Paramètre	Description
Classes	Nombre de classes
Limite inférieure	Limite inférieure de l'histogramme
Limite supérieure	Limite supérieure de l'histogramme

### Largeur à mi-hauteur

Calcule la largeur à mi-hauteur (LMH) du signal sélectionné, en utilisant l'une des méthodes suivantes :

Méthode	Description
Passage par zéro	Recherche les passages par zéro du signal après avoir centré son amplitude autour de zéro
Gauss	Ajuste les données à un modèle gaussien en utilisant un algorithme de moindres carrés
Lorentz	Ajuste les données à un modèle lorentzien en utilisant un algorithme de moindres carrés
Voigt	Ajuste les données à un modèle de Voigt en utilisant un algorithme de moindres carrés

	min(y)	max(y)	$\langle y \rangle$	$\sigma(y)$	$\Sigma(y)$	$\int y dx$
s000	7.6946e-23	0.398862	0.0499	0.107641	24.95	1
s000 ROI00	1.1479e-22	0.398862	0.0501004	0.10781	12.475	0.492007

Format    Resize    ☒ Background color

Close

FIG. 21 – Exemple de tableau récapitulatif de statistiques : chaque ligne est associée à une ROI (à l'exception de la première qui correspond aux statistiques calculées sur la totalité des données).

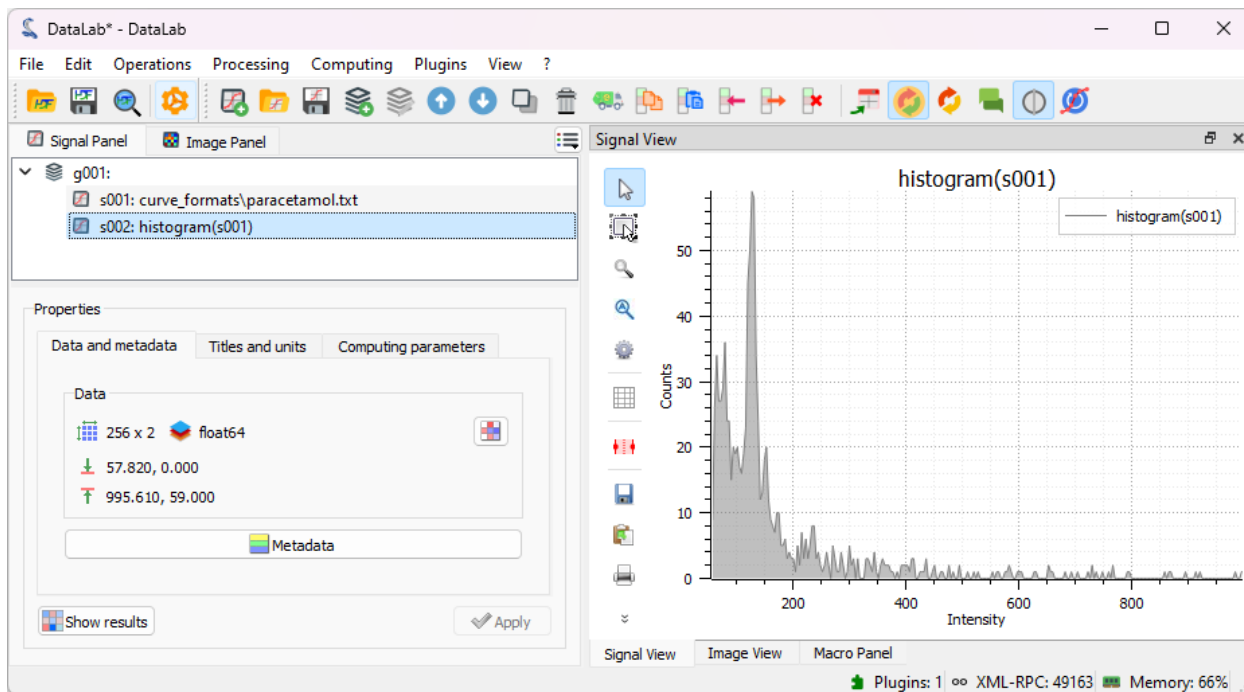


FIG. 22 – Exemple d'histogramme.

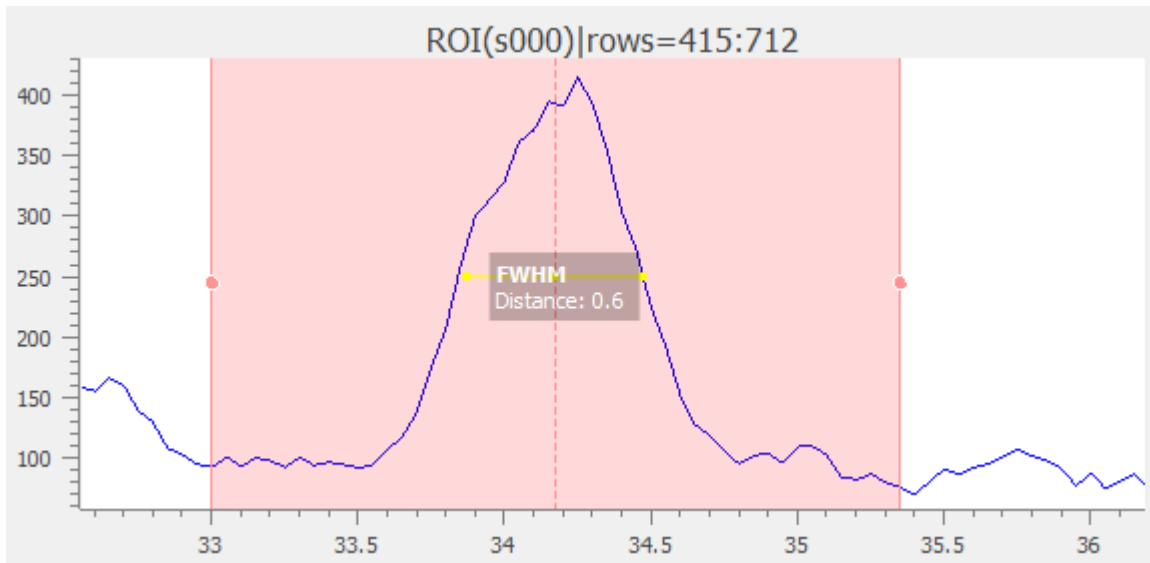


FIG. 23 – Le résultat du calcul est affiché sous la forme d'un segment annoté.

### Largeur à $1/e^2$

Réalise l'ajustement des données à une gaussienne en utilisant un algorithme de moindres carrés. Calcule ensuite la largeur à  $1/e^2$  du modèle d'ajustement.

**Note :** Les résultats de calcul scalaires sont systématiquement stockés dans les métadonnées. Les métadonnées sont attachées au signal et sérialisées avec ce dernier par exemple lors de l'export d'une session de DataLab vers un fichier HDF5.

### Valeurs de X aux min/max

Calcule les valeurs de X aux minima et maxima du signal sélectionné.

### Détection de pics

Crée un signal à partir de la détection automatique des pics de chaque signal sélectionné :

### Fréquence et période d'échantillonnage

Calcule la fréquence et la période d'échantillonnage du signal sélectionné.

**Avertissement :** Cette fonctionnalité suppose que les valeurs de X sont régulièrement espacées.

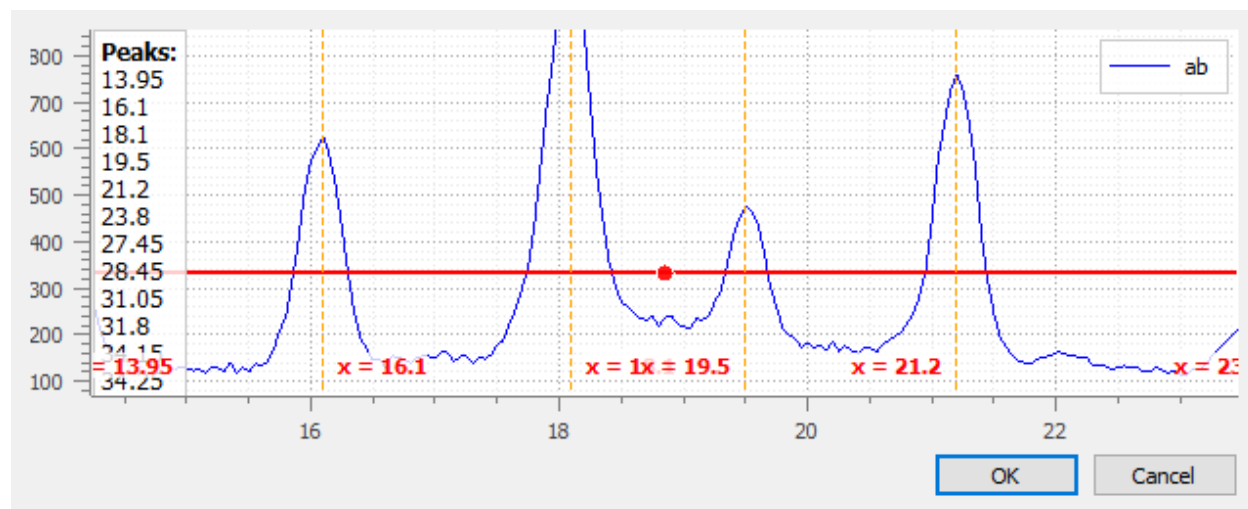


FIG. 24 – Boîte de dialogue de détection de pics : le seuil de détection est ajustable en déplaçant le curseur horizontal, les pics sont détectés automatiquement (des marqueurs verticaux indiquent les pics détectés avec leur position)

### Paramètres dynamiques

Calcule les paramètres dynamiques suivants sur le signal sélectionné :

Paramètre	Description
f	Fréquence (ajustement sinusoïdal)
ENOB	Bits effectifs
SNR	Rapport signal/bruit
SINAD	Rapport signal/bruit et distorsion
THD	Distorsion harmonique totale
SFDR	Dynamique sans distorsion

### Bande passante à -3 dB

En prenant l'hypothèse que le signal est une réponse de filtre, calcule la bande passante à -3 dB en trouvant la plage de fréquences où le signal est supérieur à -3 dB.

**Avertissement :** Cette fonctionnalité suppose que le signal est une réponse de filtre, déjà exprimée en dB.

### Contraste

Calcule le contraste du signal sélectionné.

Le contraste est défini comme le rapport de la différence et de la somme des valeurs maximale et minimale :

$$\text{Contrast} = \frac{\max(y) - \min(y)}{\max(y) + \min(y)}$$

**Note :** Cette fonctionnalité suppose que le signal est un profil extrait d'une image, pour laquelle le contraste a un sens. Cela justifie la définition optique du contraste.

### Afficher les résultats

Affiche les résultats de toutes les analyses effectuées sur les signaux sélectionnés. Cela affiche le même tableau que celui affiché après avoir effectué un calcul.

### Tracer les résultats

Trace les résultats des analyses effectuées sur les signaux sélectionnés, avec des axes X et Y définis par l'utilisateur (p.ex. trace la largeur à mi-hauteur en fonction de l'indice du signal).

## 2.3.6 Options d'affichage pour les signaux

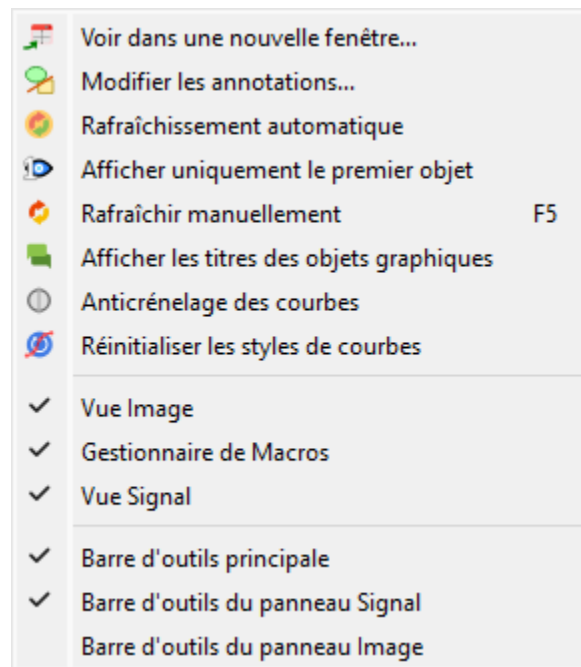


FIG. 25 – Capture d'écran du menu « Affichage ».

Lorsque le « Panneau Signal » est sélectionné, les menus et barres d'outils sont mis à jour pour fournir les actions liées aux signaux.

Le menu « Affichage » permet de visualiser le signal ou le groupe de signaux courant. Il permet également d'afficher/cacher les titres, d'activer/désactiver l'anticrénelage, ou encore de rafraîchir la visualisation.

## Voir dans une nouvelle fenêtre

Ouvre une nouvelle fenêtre pour visualiser les signaux sélectionnés.

Cette option vous permet de visualiser les signaux sélectionnés dans une fenêtre séparée, dans laquelle vous pouvez visualiser les données plus confortablement (par exemple, en maximisant la fenêtre) et vous pouvez également annoter les données.

Lorsque vous cliquez sur le bouton « Annotations » dans la barre d'outils de la nouvelle fenêtre, le mode d'édition des annotations est activé (voir la section « Modifier les annotations » ci-dessous).

## Modifier les annotations

Ouvre une nouvelle fenêtre pour modifier les annotations.

Cette option vous permet d'afficher les signaux sélectionnés dans une fenêtre séparée, dans laquelle vous pouvez visualiser les données plus confortablement (par exemple, en maximisant la fenêtre) et ajouter ou modifier des annotations.

Les annotations sont utilisées pour ajouter des commentaires ou des étiquettes aux données, et elles peuvent être utilisées pour mettre en évidence des événements spécifiques ou des régions d'intérêt.

**Note :** Les annotations sont enregistrées dans les métadonnées du signal, elles sont donc persistantes et seront affichées chaque fois que vous visualiserez le signal.

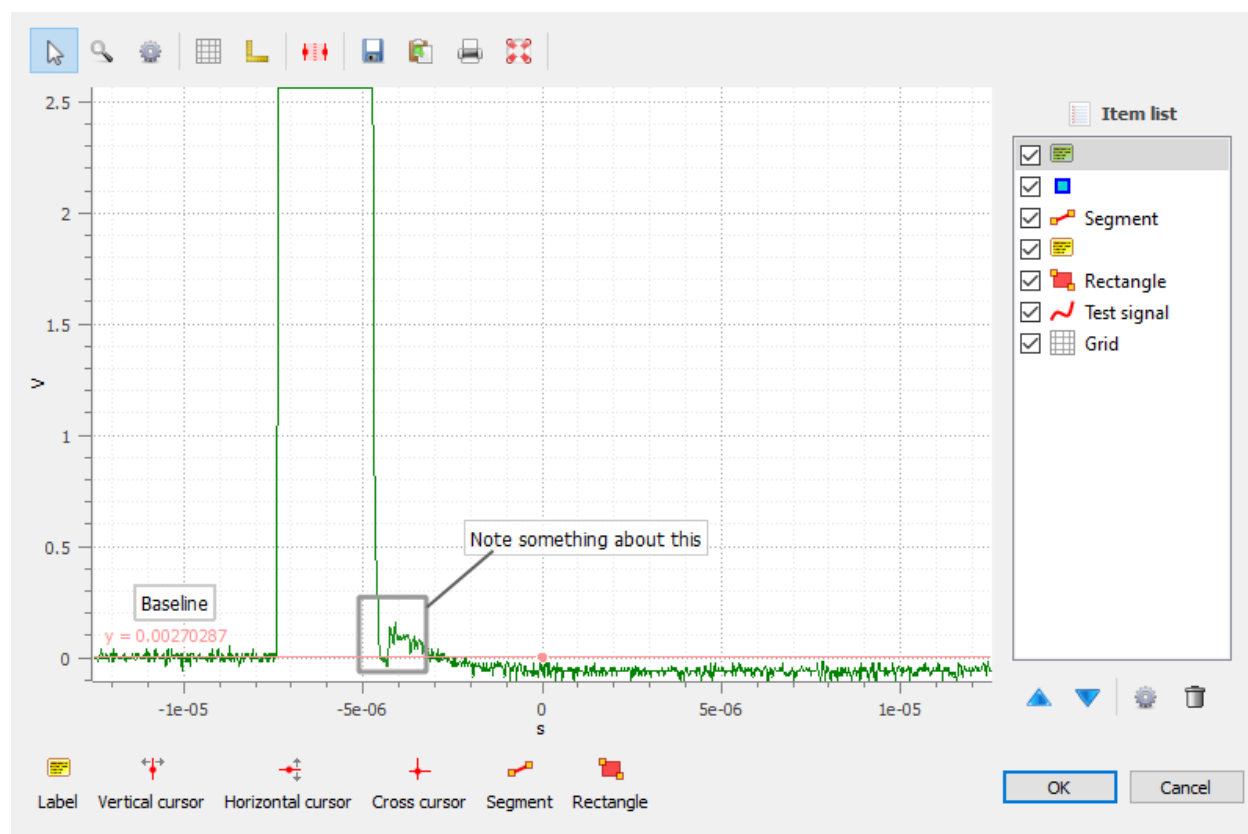


FIG. 26 – Les annotations peuvent être ajoutées dans la vue séparée.

Le mode opératoire typique pour modifier les annotations est le suivant :

1. Sélectionnez l'option « Modifier les annotations ».
2. Dans la nouvelle fenêtre, ajoutez des annotations en cliquant sur les boutons correspondants en bas de la fenêtre.
3. Personnalisez les annotations en modifiant leurs propriétés (par exemple, le texte, la couleur, la position, etc.) en utilisant l'option « Paramètres » dans le menu contextuel des annotations.
4. Quand vous avez terminé, cliquez sur le bouton « OK » pour enregistrer les annotations. Cela fermera la fenêtre et les annotations seront enregistrées dans les métadonnées du signal et seront affichées dans la fenêtre principale.

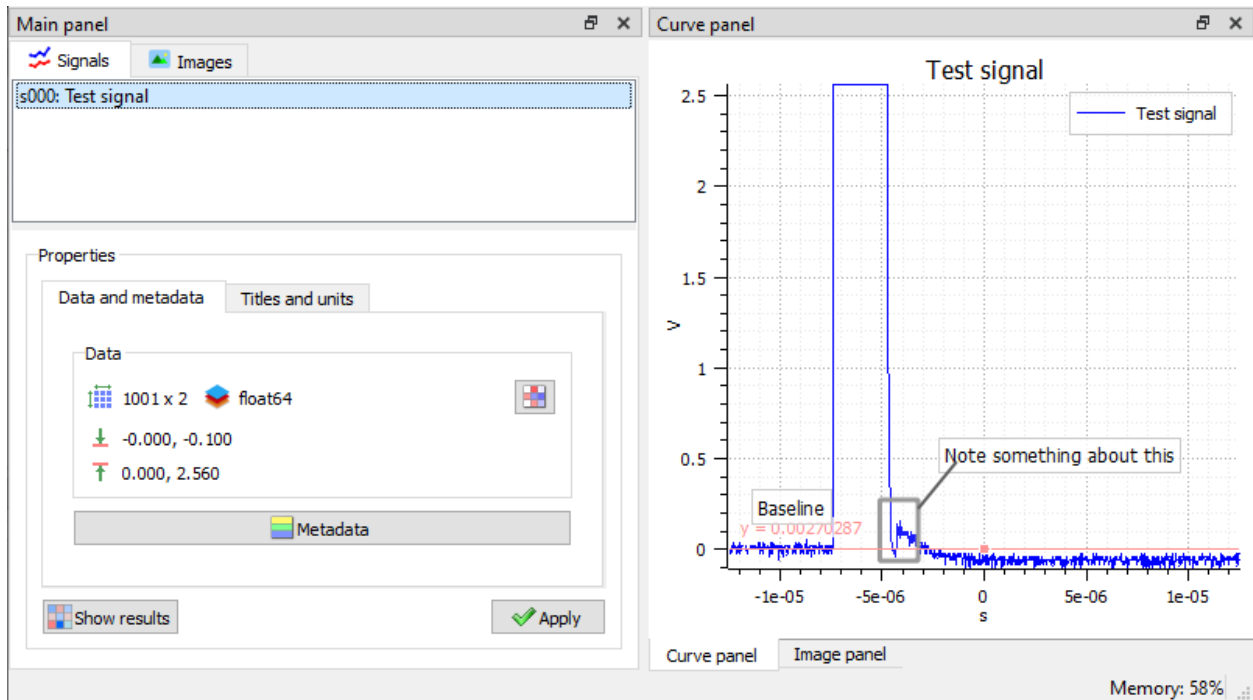


FIG. 27 – Les annotations font désormais partie des métadonnées du signal.

**Note :** Les annotations peuvent être copiées d'un signal à un autre en utilisant les fonctionnalités « copier/coller les métadonnées ».

### Rafraîchissement automatique

Rafraîchit automatiquement la visualisation quand les données changent.

- Si activé (par défaut), la visualisation est automatiquement rafraîchie quand les données changent.
- Si désactivé, la visualisation n'est pas rafraîchie tant que vous ne cliquez pas sur le bouton « Rafraîchir manuellement » dans la barre d'outils.

Même si l'algorithme de rafraîchissement est optimisé, il peut prendre du temps pour rafraîchir la visualisation quand les données changent, surtout quand le jeu de données est grand. Par conséquent, vous pouvez désactiver le rafraîchissement automatique quand vous travaillez avec des données volumineuses, et l'activer à nouveau quand vous avez terminé. Cela évitera des rafraîchissements inutiles.

## Rafraîchir manuellement

Rafraîchir la visualisation manuellement.

Cela déclenche un rafraîchissement de la visualisation. C'est utile quand le rafraîchissement automatique est désactivé, ou quand vous voulez forcer un rafraîchissement de la visualisation.

## Afficher les titres des objets graphiques

Affiche/cache les titres des objets graphiques liés aux résultats d'analyse et aux annotations.

Cette option vous permet d'afficher ou de masquer les titres des objets graphiques (par exemple, les titres des résultats d'analyse ou des annotations). Masquer les titres peut être utile lorsque vous voulez visualiser les données sans être perturbé, ou s'il y a trop de titres et qu'ils se chevauchent.

## Anticrénelage des courbes

Active/désactive l'anticrénelage sur les courbes.

L'anticrénelage rend les courbes plus lisses, mais peut aussi les rendre moins nettes.

---

**Note :** L'anticrénelage est activé par défaut.

---

**Avertissement :** L'anticrénelage peut ralentir significativement l'affichage, surtout quand on travaille avec des données volumineuses.

## Réinitialiser les styles de courbes

Réinitialise le style de courbe au début du cycle.

Lors de l'affichage de courbes, DataLab attribue automatiquement une couleur et un style de ligne à chaque courbe. Les deux paramètres sont choisis dans une liste prédéfinie de couleurs et de styles de ligne, et sont attribués de manière cyclique.

Cette entrée de menu permet de réinitialiser les styles de courbes, de sorte que la prochaine fois que vous afficherez des courbes, la première courbe sera attribuée la première couleur et le premier style de ligne de la liste prédéfinie, et la boucle recommencera à partir de là.

## 2.4 Traitement d'image

Cette section décrit les fonctionnalités spécifiques au panneau de traitement d'image. Le panneau de traitement d'image peut être sélectionné en cliquant sur l'onglet « Images » en bas à droite de la fenêtre principale de DataLab.

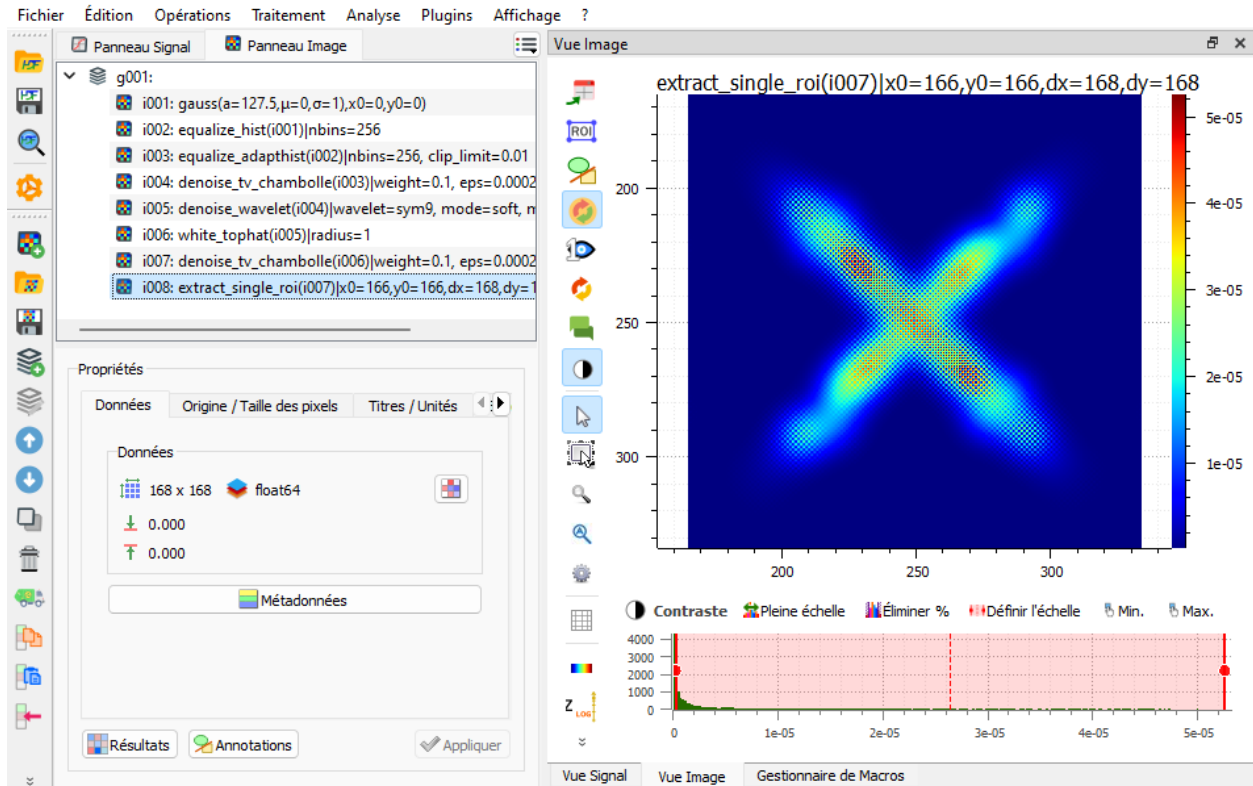


FIG. 28 – Fenêtre principale de DataLab : vue du traitement d'image

### 2.4.1 Créer, ouvrir et enregistrer des images

Cette section décrit comment créer, ouvrir et enregistrer des images (et des espaces de travail).

Lorsque le « Panneau Image » est sélectionné, les menus et barres d'outils sont mis à jour pour fournir les actions liées aux images.

Le menu « Fichier » vous permet de :

- Créer, ouvrir, enregistrer et fermer des images (voir ci-dessous).
- Enregistrer et restaurer l'espace de travail actuel ou parcourir les fichiers HDF5 (voir [Espace de travail](#)).
- Modifier les préférences de DataLab (voir [Préférences](#)).

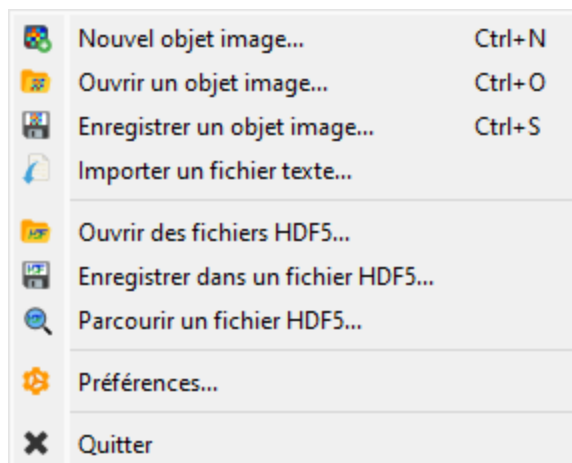


FIG. 29 – Capture d’écran du menu « Fichier ».

## Nouvelle image

Crée une image à partir de différents modèles (types de données pris en charge : uint8, uint16, int16, float32, float64) :

Modèle	Equation
Zéros	$z[i] = 0$
Vide	Données mémoire en l’état
Aléatoire	$z[i] \in [0, z_{max})$ où $z_{max}$ est la valeur maximale correspondant au type de données
Gaussienne 2D	$z = A.exp(-\frac{(\sqrt{(x-x_0)^2 + (y-y_0)^2} - \mu)^2}{2\sigma^2})$

## Ouvrir une image

Crée une image depuis l’un des types de fichiers pris en charge :

Type de fichier	Extensions
Fichiers PNG	.png
Fichiers TIFF	.tif, .tiff
Images 8bits	.jpg, .gif
Tableaux NumPy	.npy
Fichiers MAT	.mat
Fichiers texte	.txt, .csv, .asc
Fichiers Andor SIF	.sif
Fichiers Princeton Instruments SPE	.spe
Fichiers Opticks GEL	.gel
Fichiers Hammamatsu NDPI	.ndpi
Fichiers PCO Camera REC	.rec
Fichiers SPIRICON	.scor-data
Fichiers FXD	.fxd
Images Bitmap	.bmp

**Note :** DataLab prend également en charge tout format d'image pouvant être lu par la bibliothèque *imageio*, à condition que le(s) plugin(s) associé(s) soient installé(s) (voir [documentation imageio](#)) et que le type de données et la forme du tableau NumPy de sortie soient pris en charge par DataLab.

Pour ajouter un nouveau format de fichier, vous pouvez utiliser l'entrée *imageio\_formats* du fichier de configuration de DataLab. Cette entrée est formatée comme l'objet *IMAGEIO\_FORMATS* qui représente les formats pris en charge nativement :

```
cdl.config.IMAGEIO_FORMATS = (('*.gel', 'Opticks GEL'), ('*.spe', 'Princeton Instruments SPE'), ('*.ndpi', 'Hamamatsu Slide Scanner NDPI'), ('*.rec', 'PCO Camera REC'))
```

## Enregistrer l'image

Enregistre l'image sélectionnée dans l'un des types de fichier pris en charge :

## Importer un fichier texte

DataLab peut importer nativement de nombreux types de fichiers image (par exemple TIFF, JPEG, PNG, etc.). Cependant, certains formats de fichiers texte spécifiques peuvent ne pas être pris en charge. Dans ce cas, vous pouvez utiliser la fonctionnalité « Importer un fichier texte », qui vous permet d'importer un fichier texte et de le convertir en image.

Cette fonctionnalité est accessible depuis le menu « Fichier », sous l'option « Importer un fichier texte ».

Il ouvre un assistant d'importation qui vous guide tout au long du processus d'importation du fichier texte.

### Etape 1 : Sélectionner la source

La première étape consiste à sélectionner la source du fichier texte. Vous pouvez soit sélectionner un fichier de votre ordinateur, soit le presse-papiers si vous avez copié le texte à partir d'une autre application.

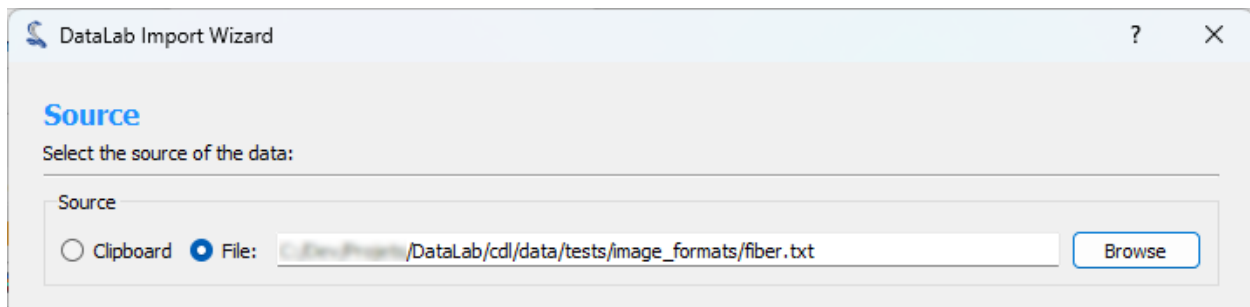


FIG. 30 – Etape 1 : Sélectionner la source

## Etape 2 : Aperçu et configuration de l'importation

La deuxième étape consiste à configurer l'importation et à prévisualiser le résultat. Vous pouvez configurer les options suivantes :

- **Délimiteur** : Le caractère utilisé pour séparer les valeurs dans le fichier texte.
- **Commentaires** : Le caractère utilisé pour indiquer que la ligne est un commentaire et doit être ignorée.
- **Lignes à sauter** : Le nombre de lignes à sauter au début du fichier.
- **Nombre maximum de lignes** : Le nombre maximum de lignes à importer. Si le fichier contient plus de lignes, elles seront ignorées.
- **Transposer** : Si coché, les lignes et les colonnes seront transposées.
- **Type de données** : Le type de données de destination des données importées.

Lorsque vous avez terminé de configurer l'importation, cliquez sur le bouton « Appliquer » pour voir le résultat.

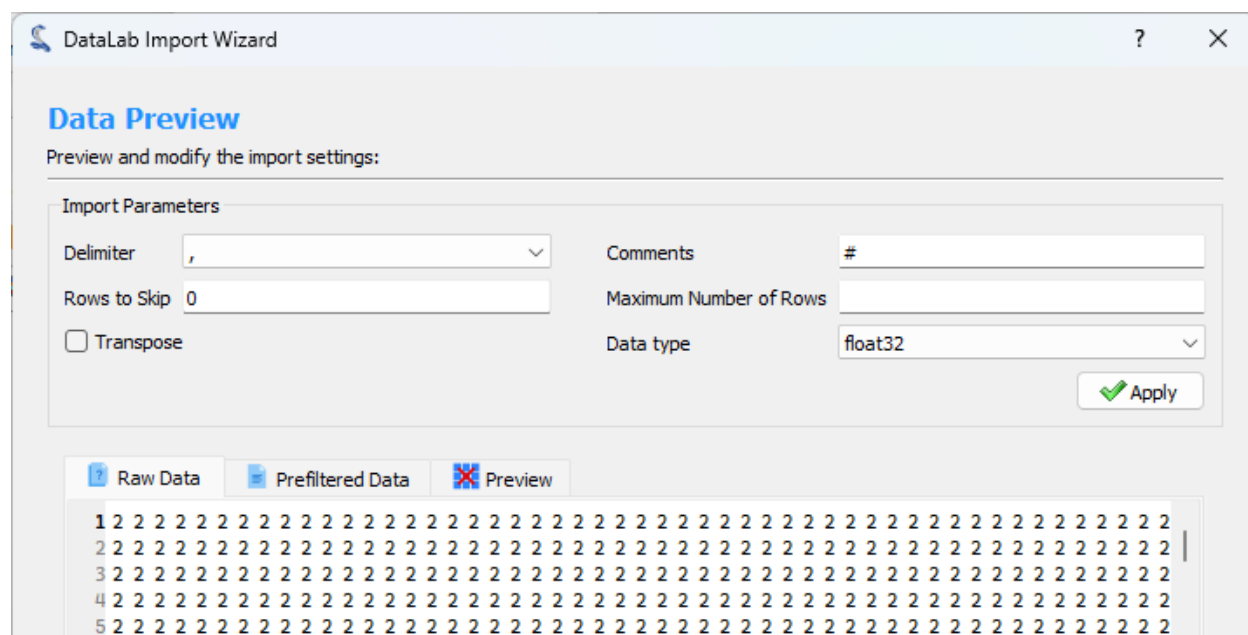


FIG. 31 – Etape 2 : Configurer l'importation

## Etape 3 : Afficher la représentation graphique

La troisième étape montre une représentation graphique des données importées. Vous pouvez utiliser le bouton « Terminer » pour importer les données dans l'espace de travail de DataLab.

### 2.4.2 Manipuler les métadonnées

Cette section décrit comment manipuler les métadonnées dans DataLab.

Le menu « Édition » vous permet d'effectuer des opérations d'édition classiques sur l'image ou le groupe d'images actuel (créer/renommer un groupe, déplacer vers le haut/vers le bas, supprimer l'image/le groupe d'images, etc.).

Il vous permet également de manipuler les métadonnées associées à l'image actuelle.

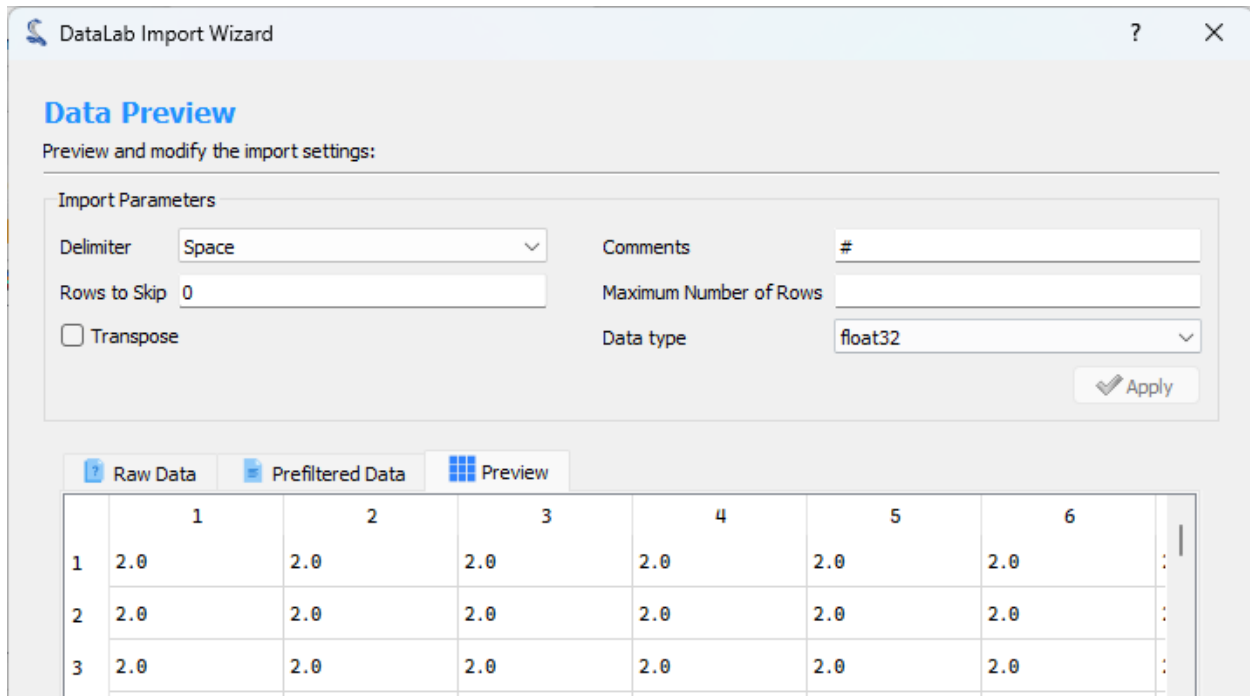


FIG. 32 – Etape 2 : Prévisualiser le résultat

### Copier/coller les métadonnées

Compte tenu du fait que les métadonnées contiennent des informations utiles sur l'image, elles peuvent être copiées et collées d'une image à une autre en sélectionnant les actions « Copier les métadonnées » et « Coller les métadonnées » dans le menu « Édition ».

Cette fonctionnalité vous permet de transférer ces informations d'une image à une autre :

- *Régions d'intérêt (ROIs)* : c'est un moyen très efficace de réutiliser la même ROI sur différentes images et de comparer facilement les résultats de l'analyse sur ces images
- Résultats de calcul, tels qu'une position de barycentre ou une détection de contour (la pertinence du transfert de ces informations dépend du contexte et il appartient à l'utilisateur d'en décider)
- Toute autre information que vous avez pu ajouter aux métadonnées d'une image

**Note :** Copier les métadonnées d'une image à une autre écrasera les métadonnées de l'image de destination (pour les clés de métadonnées communes aux deux images) ou ajoutera simplement les clés de métadonnées qui ne sont pas présentes dans l'image de destination.

### Importer/exporter les métadonnées

Les métadonnées peuvent également être importées et exportées depuis/vers un fichier JSON en utilisant les actions « Importer les métadonnées » et « Exporter les métadonnées » dans le menu « Édition ». C'est exactement la même chose que la fonctionnalité de copier/coller des métadonnées (voir ci-dessus pour plus de détails sur les cas d'utilisation de cette fonctionnalité), mais cela vous permet de sauvegarder les métadonnées dans un fichier et de les importer ultérieurement.

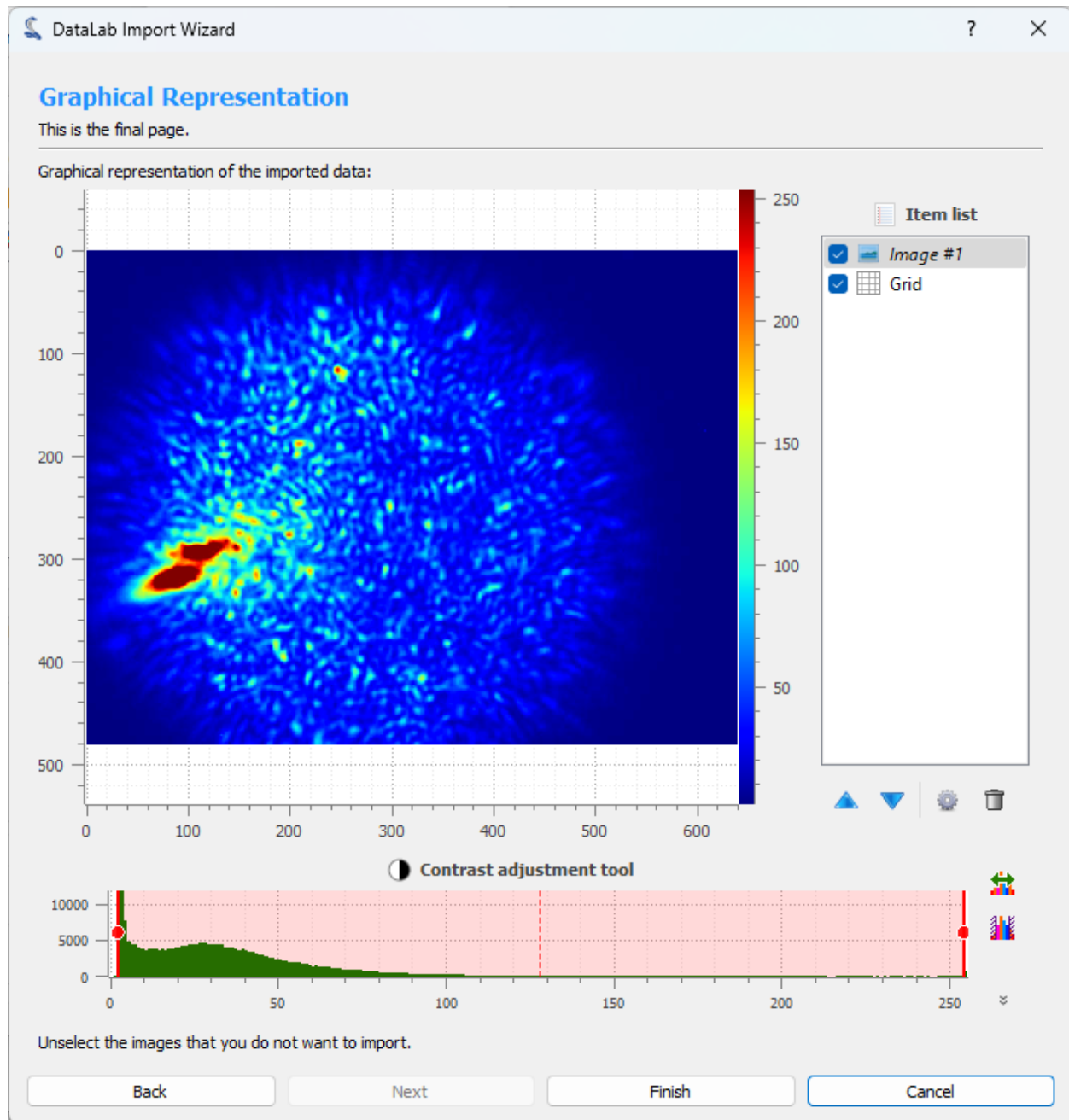


FIG. 33 – Etape 3 : Afficher la représentation graphique

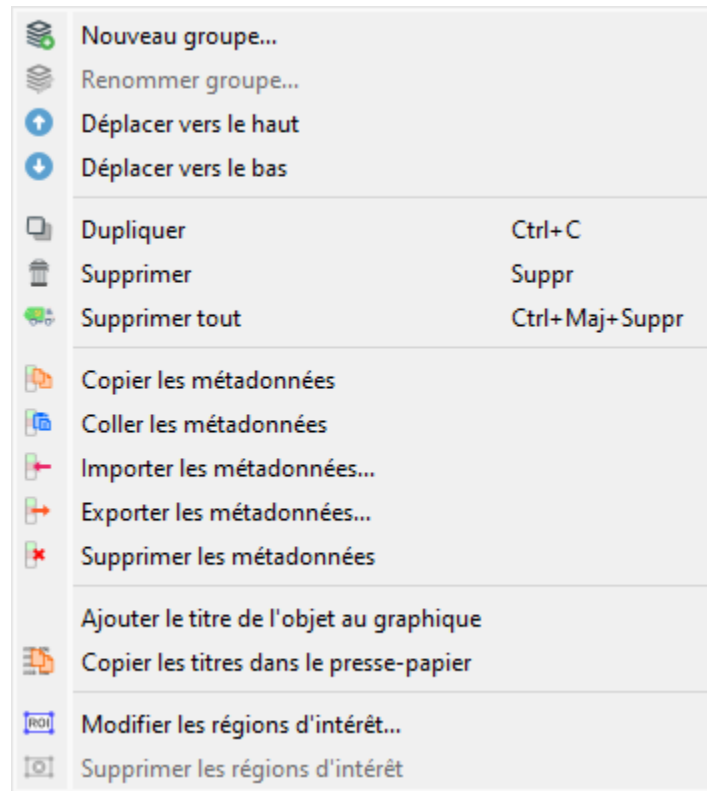


FIG. 34 – Capture d’écran du menu « Édition ».

## Supprimer les métadonnées

Lors de la suppression des métadonnées en utilisant l’action « Supprimer les métadonnées » dans le menu « Édition », vous serez invité à confirmer la suppression des régions d’intérêt (ROIs) si elles sont présentes dans les métadonnées. Après cette confirmation éventuelle, les métadonnées seront supprimées, ce qui signifie que les résultats d’analyse, les ROIs et toute autre information associée à l’image seront perdus.

## Titres des images

Les titres des images peuvent être considérés comme des métadonnées du point de vue de l’utilisateur, même s’ils ne sont pas stockés dans les métadonnées de l’image (mais dans un attribut de l’objet image).

Le menu « Édition » vous permet de :

- « Ajouter le titre de l’objet au graphique » : cette action ajoutera une étiquette en haut de l’image avec son titre, ce qui peut être utile en combinaison avec l’opération « Distribuer sur une grille » (voir *Opérations sur les images*) pour identifier facilement les images.
- « Copier les titres dans le presse-papiers » : cette action copiera les titres des images sélectionnées dans le presse-papiers, ce qui peut être utile pour les coller dans un éditeur de texte ou dans un tableur.

Exemple du contenu du presse-papiers :

```
g001:
  s001: lorentz(a=1,sigma=1,mu=0,ymin=0)
  s002: derivative(s001)
  s003: wiener(s002)
```

(suite sur la page suivante)

(suite de la page précédente)

```
g002: derivative(g001)
      s004: derivative(s001)
      s005: derivative(s002)
      s006: derivative(s003)
g003: fft(g002)
      s007: fft(s004)
      s008: fft(s005)
      s009: fft(s006)
```

## Régions d'intérêt (ROI)

Les régions d'intérêt (ROI) sont des zones d'image définies par l'utilisateur pour effectuer des opérations spécifiques, des traitements ou des analyses sur elles.

Les ROI sont prises en compte dans presque toutes les fonctionnalités de calcul de DataLab :

- Les fonctionnalités du menu « Opérations » sont appliquées uniquement sur la ROI si elle est définie (sauf si l'opération modifie la forme des données - comme l'opération de redimensionnement - ou la taille des pixels - comme l'opération de binning).
- Les actions du menu « Traitement » sont effectuées uniquement sur la ROI si elle est définie (sauf si le type de données du signal de destination est différent de celui de la source, comme dans les fonctionnalités d'analyse de Fourier ou comme dans les opérations de seuillage).
- Les actions du menu « Analyse » sont effectuées uniquement sur la ROI si elle est définie.

---

**Note :** Les ROI sont stockées en tant que métadonnées et sont donc attachées à l'image.

---

Le menu « Édition » vous permet de :

- « Modifier les régions d'intérêt » : ouvre une boîte de dialogue pour gérer les ROI associées à l'image sélectionnée (ajouter, supprimer, déplacer, redimensionner, etc.). La boîte de dialogue de définition des ROI est exactement la même que l'extraction des ROI (voir ci-dessous).
- « Supprimer les régions d'intérêt » : supprime toutes les ROI définies pour les images sélectionnées.

### 2.4.3 Opérations sur les images

Cette section décrit les opérations qui peuvent être effectuées sur les images.

**Voir aussi :**

*Traitement des images* pour plus d'informations sur les fonctionnalités de traitement d'image, ou *Analyse sur les images* pour des informations sur les fonctionnalités d'analyse des images.

Lorsque le « Panneau Image » est sélectionné, les menus et barres d'outils sont mis à jour pour fournir les actions liées aux images.

Le menu « Opérations » permet d'effectuer diverses opérations sur l'image ou le groupe d'images courant. Il permet également d'extraire des profils, de distribuer des images sur une grille, ou de redimensionner des images.

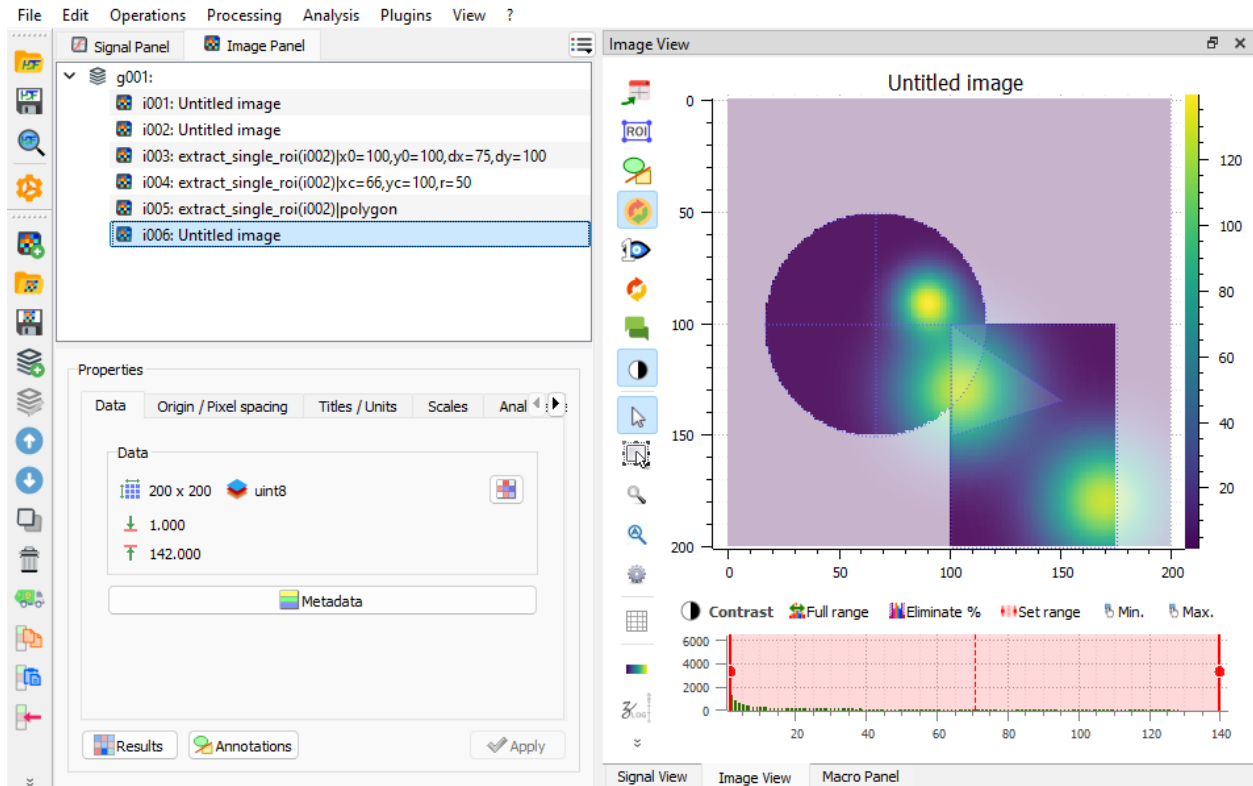


FIG. 35 – Une image avec une ROI.

## Opérations arithmétiques de base

Opération	Description
Somme	$z_M = \sum_{k=0}^{M-1} z_k$
Moyenne	$z_M = \frac{1}{M} \sum_{k=0}^{M-1} z_k$
Soustraction	$z_2 = z_1 - z_0$
Soustraction quadratique	$z_2 = \frac{z_1 - z_0}{\sqrt{2}}$
Produit	$z_M = \prod_{k=0}^{M-1} z_k$
Division	$z_2 = \frac{z_1}{z_0}$

## Opérations avec une constante

Crée une image à partir d'une opération avec une constante sur chaque image sélectionnée :

Opération	Equation
Addition	$z_k = z_{k-1} + conv(c)$
Soustraction	$z_k = z_{k-1} - conv(c)$
Multiplication	$z_k = conv(z_{k-1} \times c)$
Division	$z_k = conv\left(\frac{z_{k-1}}{c}\right)$

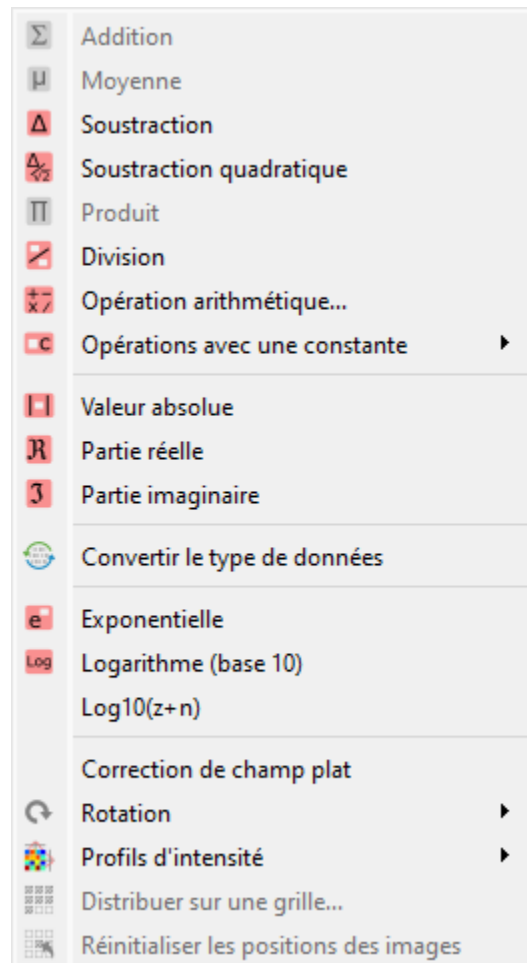


FIG. 36 – Capture d'écran du menu « Opérations ».

où  $c$  est la valeur constante et *conv* est la fonction de conversion qui gère la conversion du type de données (en conservant le même type de données que l'image d'entrée).

## Parties réelles et imaginaires

Opération	Description
Valeur absolue	$z_k =  z_{k-1} $
Partie réelle	$z_k = \Re(z_{k-1})$
Partie imaginaire	$z_k = \Im(z_{k-1})$

## Conversion du type de données

L'action « Convertir le type de données » permet de convertir le type de données des images sélectionnées. Pour les types de données entiers, la conversion est effectuée en rognant les valeurs à la plage de nouveaux types de données avant de convertir effectivement le type de données. Pour les types de données à virgule flottante, la conversion est directe.

**Note :** La conversion du type de données utilise la fonction `cdl.algorithms.datatypes.clip_astype()` qui repose sur la fonction `numpy.ndarray.astype()` avec les paramètres par défaut (*casting="unsafe"*).

## Fonctions mathématiques de base

Fonction	Description
Exponentielle	$z_k = \exp(z_{k-1})$
Logarithme (base 10)	$z_k = \log_{10}(z_{k-1})$
Log10(z+n)	$z_k = \log_{10}(z_{k-1} + n)$ (avoid Log10(0) on image background)

## Autres opérations

### Correction de champ plat

Calcule la correction de champ plat à partir des **deux** images sélectionnées :

$$z_1 = \begin{cases} \frac{z_0}{z_f} \cdot \overline{z_f} & \text{if } z_0 > z_{threshold} \\ z_0 & \text{otherwise} \end{cases}$$

où  $z_0$  est l'image brute,  $z_f$  est l'image d'homogénéité,  $z_{threshold}$  est un seuil ajustable et  $\overline{z_f}$  est la valeur moyenne de l'image d'homogénéité :

$$\overline{z_f} = \frac{1}{N_{row} \cdot N_{col}} \cdot \sum_{i=0}^{N_{row}} \sum_{j=0}^{N_{col}} z_f(i, j)$$

**Note :** L'image brute et l'image d'homogénéité sont supposées avoir déjà été corrigées par soustraction d'image de noir.

## Rotation

Crée une image qui est le résultat de la rotation (90°, 270° ou angle arbitraire) ou de l'inversion (horizontale ou verticale) des données de l'image sélectionnée.

## Profils d'intensité

### Profil rectiligne

Extraire un profil horizontal ou vertical de chaque image sélectionnée et créer un nouveau signal à partir de chacun de ces profils.

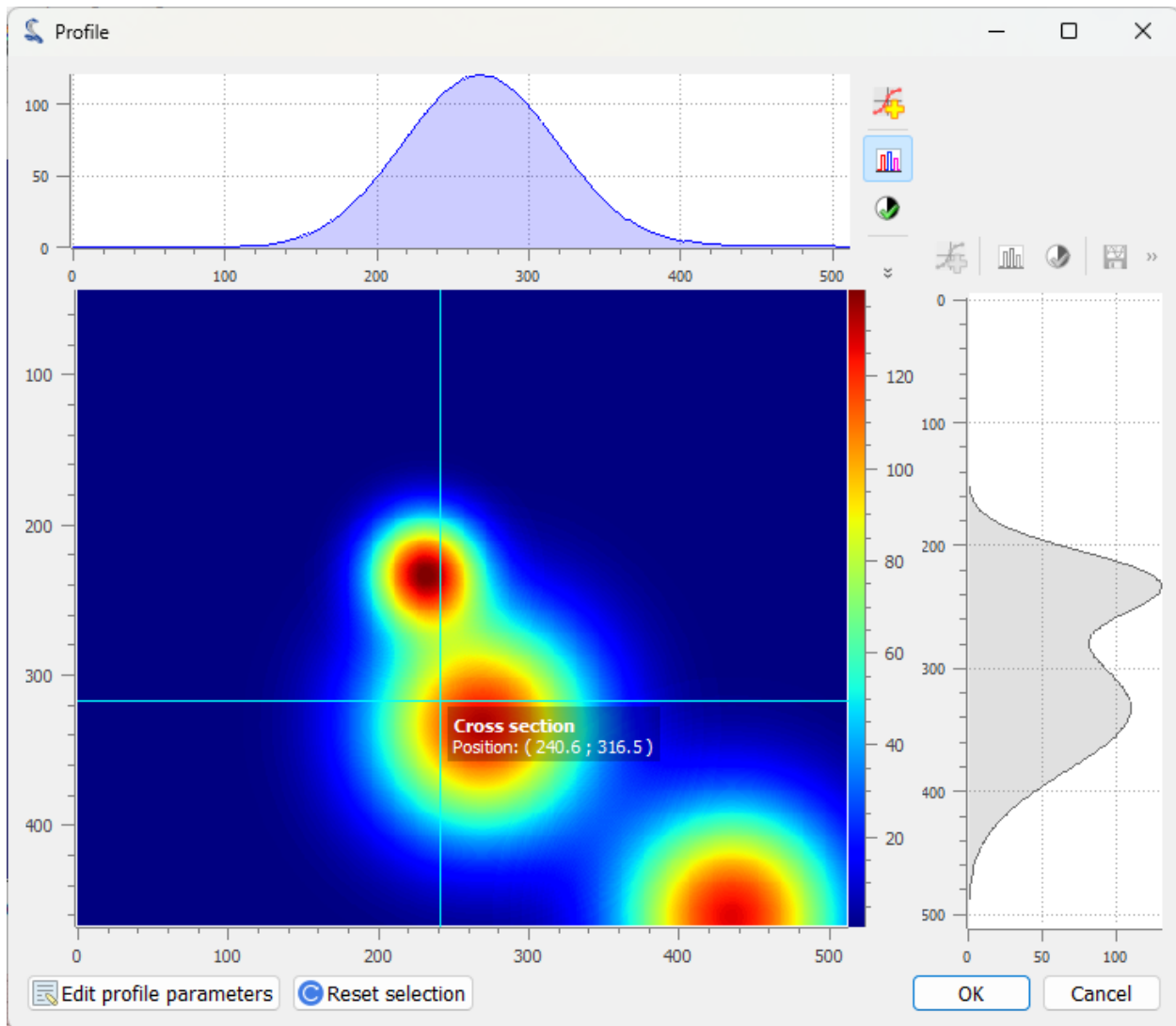


FIG. 37 – Boîte de dialogue d'extraction de profil. Les paramètres peuvent être également définis manuellement (bouton « Editer les paramètres du profil »).

### Profil le long d'un segment

Extraire un profil le long d'un segment de chaque image sélectionnée et créer un nouveau signal à partir de chacun de ces profils.

### Profil moyen

Extraire un profil horizontal ou vertical moyenné sur une zone rectangulaire de chaque image sélectionnée et créer un nouveau signal à partir de chacun de ces profils.

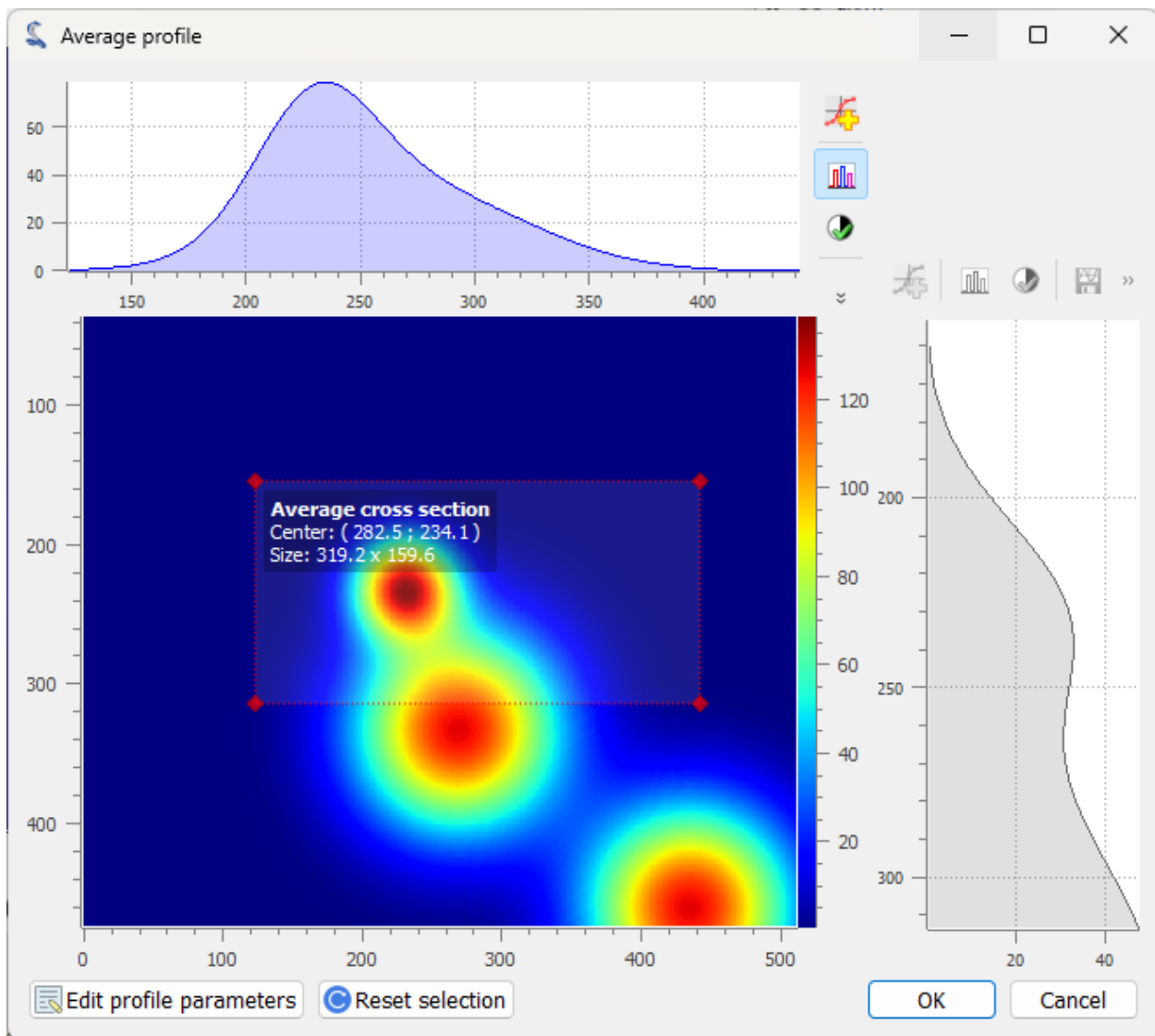


FIG. 38 – Boîte de dialogue d'extraction de profil moyen : la zone est définie par un rectangle. Les paramètres peuvent être également définis manuellement (bouton « Editer les paramètres du profil »).

### Extraire un profil radial

Extraire un profil radial de chaque image sélectionnée et créer un nouveau signal à partir de ces profils.

Les paramètres suivants sont disponibles :

Paramètre	Description
Centre	Centre autour duquel le profil radial est calculé : centre de masse, centre de l'image, ou défini par l'utilisateur
X	Coordonnée X du centre (si défini par l'utilisateur), en pixels
Y	Coordonnée Y du centre (si défini par l'utilisateur), en pixels

## Distribuer les images sur une grille

Fonctionnalité	Description
Distribuer sur une grille	Distribuer les images sélectionnées sur une grille régulière
Réinitialiser les positions	Réinitialiser les positions des images sélectionnées aux coordonnées de la première image (x0, y0)

### 2.4.4 Traitement des images

Cette section décrit les fonctionnalités de traitement d'image disponibles dans DataLab.

**Voir aussi :**

*Opérations sur les images* pour plus d'informations sur les opérations qui peuvent être effectuées sur les images, ou *Analyse sur les images* pour des informations sur les fonctionnalités d'analyse des images.

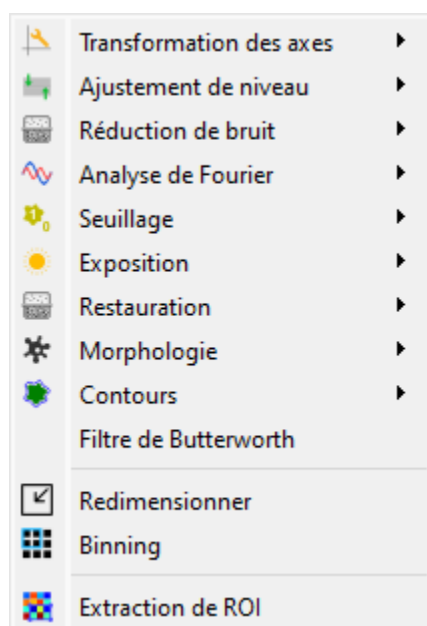


FIG. 39 – Capture d'écran du menu « Traitement ».

Lorsque le « Panneau Image » est sélectionné, les menus et barres d'outils sont mis à jour pour fournir les actions liées aux images.

Le menu « Traitement » permet d'effectuer divers traitements sur l'image ou le groupe d'images courant : il permet d'appliquer des filtres, de corriger l'exposition, de réduire le bruit, d'effectuer des opérations morphologiques, etc.

## Transformation des axes

### Étalonnage linéaire

Crée une image à partir de l'étalonnage linéaire (par rapport à l'axe des Z) de chaque image sélectionnée.

Paramètre	Étalonnage linéaire
Axe des Z	$z_1 = a.z_0 + b$

### Permuter les axes X/Y

Crée une image à partir des données inversées X/Y de l'image sélectionnée.

### Ajustement des niveaux

#### Normalisation

Crée une image à partir de la normalisation de chaque image sélectionnée par maximum, amplitude, somme, énergie ou RMS :

Normalisation	Equation
Maximum	$z_1 = \frac{z_0}{z_{max}}$
Amplitude	$z_1 = \frac{z_0}{z_{max} - z_{min}}$
Aire	$z_1 = \frac{z_0}{\sum_{i=0}^{N-1} z_i}$
Energie	$z_1 = \frac{z_0}{\sqrt{\sum_{n=0}^N  z_0[n] ^2}}$
RMS	$z_1 = \frac{z_0}{\sqrt{\frac{1}{N} \sum_{n=0}^N  z_0[n] ^2}}$

### Ecrêtage

Applique un écrêtage sur chaque image sélectionnée.

### Soustraction d'offset

Crée une image à partir du résultat d'une correction d'offset sur chaque image sélectionnée. Cette opération est réalisée en soustrayant la valeur de fond de l'image, qui est estimée par la valeur moyenne d'une zone rectangulaire définie par l'utilisateur.

## Réduction de bruit

Crée une image à partir du résultat d'un débruitage sur chaque image sélectionnée.

Les filtres suivants sont disponibles :

Filtre	Formule/implémentation
Filtre gaussien	<code>scipy.ndimage.gaussian_filter</code>
Moyenne mobile	<code>scipy.ndimage.uniform_filter</code>
Médiane mobile	<code>scipy.ndimage.median_filter</code>
Filtre de Wiener	<code>scipy.signal.wiener</code>

## Analyse de Fourier

Crée une image à partir du résultat d'une analyse de Fourier sur chaque image sélectionnée.

Les fonctions suivantes sont disponibles :

Fonction	Description	Formule/implémentation
FFT	Transformée de Fourier rapide	<code>numpy.fft.fft2</code>
FFT inverse	Transformée de Fourier rapide inverse	<code>numpy.fft.ifft2</code>
Spectre d'amplitude	Optionnel : utiliser une échelle logarithmique (dB)	$z_1 =  FFT(z_0) $ ou $z_1 = 20 \log_{10}( FFT(z_0) )$ (dB)
Spectre de phase		$z_1 = \angle(FFT(z_0))$
Densité spectrale de puissance	Optionnel : utiliser une échelle logarithmique (dB)	$z_1 =  FFT(z_0) ^2$ ou $z_1 = 10 \log_{10}( FFT(z_0) ^2)$ (dB)

**Note :** La FFT et la FFT inverse sont effectuées avec décalage de fréquence si l'option est activée dans les paramètres de DataLab (voir [Préférences](#)).

## Seuillage

Crée une image à partir du résultat d'un seuillage sur chaque image, éventuellement basé sur des paramètres définis par l'utilisateur (« Seuillage paramétrique »).

Les paramètres suivants sont disponibles lors de la sélection de « Seuillage paramétrique » :

Paramètre	Description
Méthode de seuillage	La méthode de seuillage à utiliser (voir le tableau ci-dessous)
Classes	Nombre de classes pour le calcul de l'histogramme
Valeur	Valeur de seuil
Opération	Opération à appliquer (> ou <)

Les méthodes de seuillage suivantes sont disponibles :

Méthode	Implémentation
Manuel	Seuillage manuel (paramètres définis par l'utilisateur)
ISODATA	<code>skimage.filters.threshold_isodata</code>
Li	<code>skimage.filters.threshold_li</code>
Moyenne	<code>skimage.filters.threshold_mean</code>
Minimum	<code>skimage.filters.threshold_minimum</code>
Otsu	<code>skimage.filters.threshold_otsu</code>
Triangle	<code>skimage.filters.threshold_triangle</code>
Yen	<code>skimage.filters.threshold_yen</code>

**Note :** L'option « Toutes les méthodes de seuillage » permet d'appliquer toutes les méthodes de seuillage à la même image. Combinée avec l'option « distribuer sur une grille », cela permet de comparer les différentes méthodes de seuillage sur la même image.

## Exposition

Crée une image à partir du résultat d'une correction d'exposition sur chaque image sélectionnée.

Les fonctions suivantes sont disponibles :

Fonction	Implémentation	Commentaires
Correction gamma	<code>skimage.exposure.adjust_gamr</code>	
Correction logarithmique	<code>skimage.exposure.adjust_log</code>	
Correction sigmoïde	<code>skimage.exposure.adjust_sigm</code>	
Egalisation d'histogramme	<code>skimage.exposure.equalize_hist</code>	
Egalisation d'histogramme adaptative	<code>skimage.exposure.equalize_adapt</code>	Algorithme CLAHE (Contrast Limited Adaptive Histogram Equalization)
Ajustement des niveaux	<code>skimage.exposure.rescale_intensity</code>	Réduit ou étend la plage de répartition des niveaux de l'image

## Restauration

Crée une image à partir du résultat d'une restauration sur chaque image sélectionnée.

Les fonctions suivantes sont disponibles :

Fonction	Implémentation	Commentaires
Débruitage par variation totale	<code>ski-image.restoration.denoise_tv</code>	
Débruitage par filtre bilatéral	<code>ski-image.restoration.denoise_bi</code>	
Débruitage par ondelettes	<code>ski-image.restoration.denoise_w</code>	
Débruitage par Top-Hat	<code>ski-image.morphology.white_tophat</code>	Débruite l'image en soustrayant sa transformation Top-Hat

**Note :** L'option « Toutes les méthodes de débruitage » permet d'appliquer toutes les méthodes de débruitage à la même image. Combinée avec l'option « distribuer sur une grille », cela permet de comparer les différentes méthodes de débruitage sur la même image.

## Morphologie

Crée une image à partir du résultat d'opérations morphologiques sur chaque image sélectionnée, en utilisant un disque comme empreinte.

Les fonctions suivantes sont disponibles :

Fonction	Implémentation
Top-Hat (disque)	<code>skimage.morphology.white_tophat</code>
Top-Hat dual (disque)	<code>skimage.morphology.black_tophat</code>
Erosion (disque)	<code>skimage.morphology.erosion</code>
Dilatation (disque)	<code>skimage.morphology.dilation</code>
Ouverture (disque)	<code>skimage.morphology.opening</code>
Fermeture (disque)	<code>skimage.morphology.closing</code>

**Note :** L'option « Toutes les opérations morphologiques » permet d'appliquer toutes les opérations morphologiques à la même image. Combinée avec l'option « distribuer sur une grille », cela permet de comparer les différentes opérations morphologiques sur la même image.

## Contours

Crée une image à partir du résultat d'un filtrage de contours sur chaque image sélectionnée.

Les fonctions suivantes sont disponibles :

Fonction	Implémentation
Filtre de Roberts	<code>skimage.filters.roberts</code>
Filtre de Prewitt	<code>skimage.filters.prewitt</code>
Filtre de Prewitt (horizontal)	<code>skimage.filters.prewitt_h</code>
Filtre de Prewitt (vertical)	<code>skimage.filters.prewitt_v</code>
Filtre de Sobel	<code>skimage.filters.sobel</code>
Filtre de Sobel (horizontal)	<code>skimage.filters.sobel_h</code>
Filtre de Sobel (vertical)	<code>skimage.filters.sobel_v</code>
Filtre de Scharr	<code>skimage.filters.scharr</code>
Filtre de Scharr (horizontal)	<code>skimage.filters.scharr_h</code>
Filtre de Scharr (vertical)	<code>skimage.filters.scharr_v</code>
Filtre de Farid	<code>skimage.filters.farid</code>
Filtre de Farid (horizontal)	<code>skimage.filters.farid_h</code>
Filtre de Farid (vertical)	<code>skimage.filters.farid_v</code>
Filtre de Laplace	<code>skimage.filters.laplace</code>
Filtre de Canny	<code>skimage.feature.canny</code>

**Note :** L'option « Tous les filtres de contours » permet d'appliquer tous les algorithmes de filtrage de contours à la même image. Combinée avec l'option « distribuer sur une grille », cela permet de comparer les différents filtres de contours sur la même image.

## Filtre de Butterworth

Calcule le résultat d'un filtre de Butterworth sur l'image (implémentation basée sur `skimage.filters.butterworth`)

## Redimensionner

Crée une image qui est le résultat du redimensionnement de chaque image sélectionnée.

## Binning

Regroupe des pixels adjacents de l'image en un seul pixel (somme, moyenne, médiane, minimum ou maximum de la valeur des pixels adjacents).

## Extraction de ROI

Crée une image à partir d'une région d'intérêt (ROI) définie par l'utilisateur.

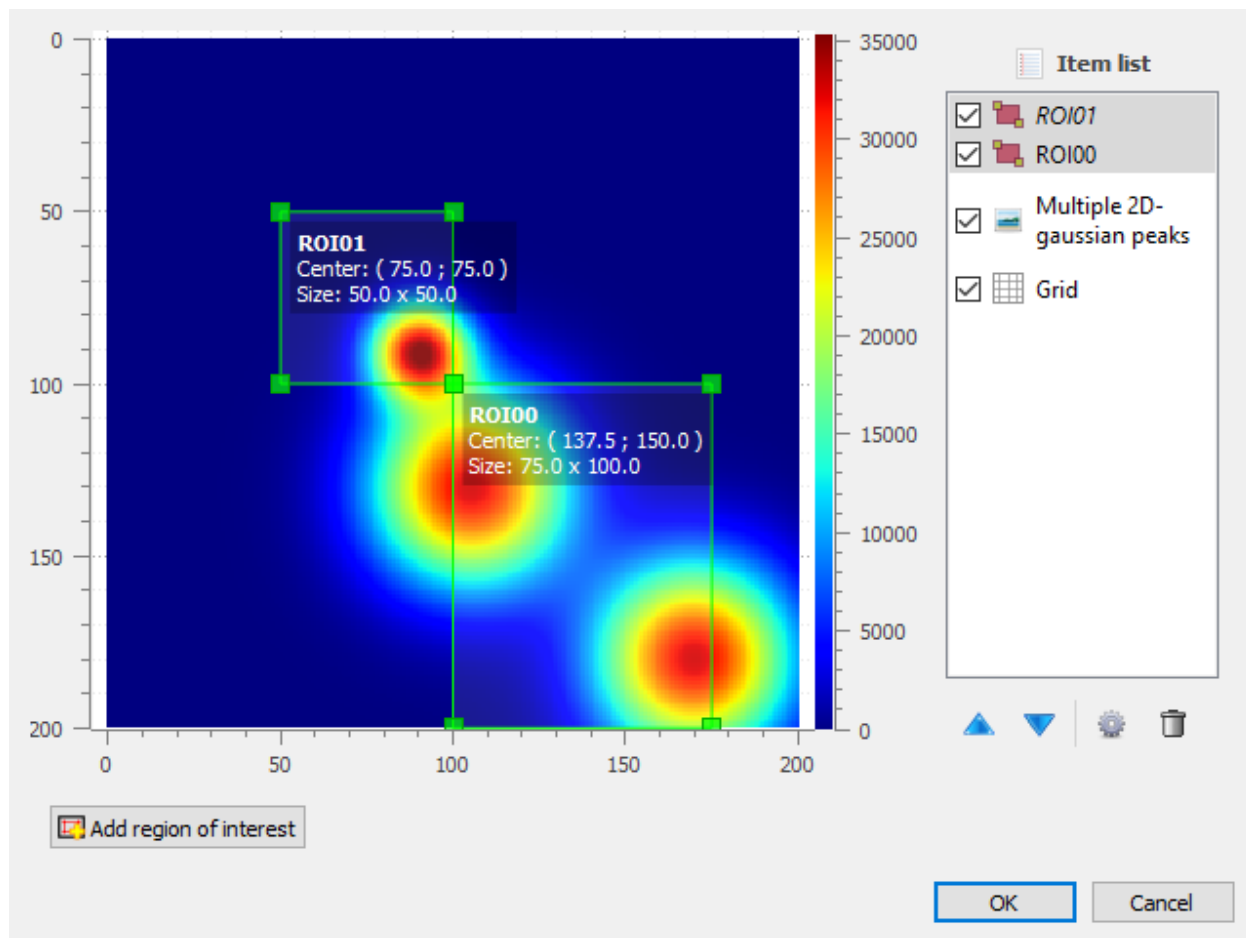


FIG. 40 – Boîte de dialogue d'extraction de ROI : la région d'intérêt (ROI) est définie en ajustant la position et la taille du rectangle de sélection.

## 2.4.5 Analyse sur les images

Cette section décrit les fonctionnalités d'analyse d'images disponibles dans DataLab.

**Voir aussi :**

*Opérations sur les images* pour plus d'informations sur les opérations pouvant être effectuées sur les images, ou *Traitement des images* pour des informations sur les fonctionnalités de traitement des images.

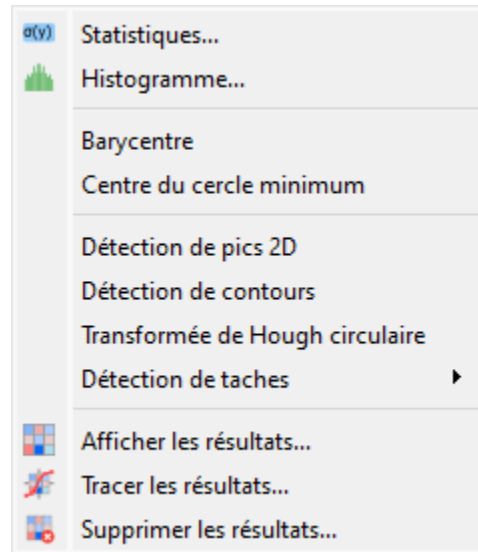


FIG. 41 – Capture d'écran du menu Analyse ».

Lorsque le « Panneau Image » est sélectionné, les menus et barres d'outils sont mis à jour pour fournir les actions liées aux images.

Le menu Analyse » permet d'effectuer divers calculs sur l'image courante ou sur un groupe d'images. Il permet également de calculer des statistiques, le barycentre, de détecter des pics, des contours, etc.

---

**Note :** Dans le vocabulaire de DataLab, une « analyse » est une fonctionnalité de calcul d'un résultat scalaire à partir d'une image. Ce résultat est stocké sous la forme de métadonnées, et donc attaché à l'image. C'est différent d'un « traitement » qui crée une nouvelle image à partir d'une image existante.

---

### Statistiques

Calcule des statistiques sur les images sélectionnées et affiche un tableau récapitulatif.

	min(z)	max(z)	<z>	$\sigma(z)$	$\Sigma(z)$	SNR(z)
i000	0	32754	512.558	2852.36	1.28139e+08	5.56494
i000 ROI00	0	32754	512.558	2852.36	6.40697e+07	5.56494
i000 ROI01	0	0	0	0	0	nan
i000 ROI02	0	32754	512.558	2852.36	3.20349e+07	5.56494

Format    Resize    ☒ Background color

Close

FIG. 42 – Exemple de tableau récapitulatif de statistiques : chaque ligne est associée à une ROI (à l'exception de la première qui correspond aux statistiques calculées sur la totalité des données).

## Histogramme

Calcule l'histogramme de l'image sélectionnée et l'affiche dans le panneau Signal.

Paramètres :

Paramètre	Description
Classes	Nombre de classes
Limite inférieure	Limite inférieure de l'histogramme
Limite supérieure	Limite supérieure de l'histogramme

## Barycentre

Calcule le barycentre en utilisant une méthode basée sur la transformée de Fourier (telle que décrite dans [Weisshaar et al.](#)). Cette méthode présente l'avantage d'être peu sensible au bruit de fond.

## Centre du cercle minimum

Calcule le contour circulaire entourant les valeurs de l'image au-delà d'un seuil (moitié du maximum de l'image).

## Détection de pics 2D

Détecte automatiquement des pics sur une image en utilisant un algorithme basé sur des filtres minimum-maximum.

**Voir aussi :**

Voir *Détection de pics 2D* pour plus de détails sur l'algorithme et les paramètres associés.

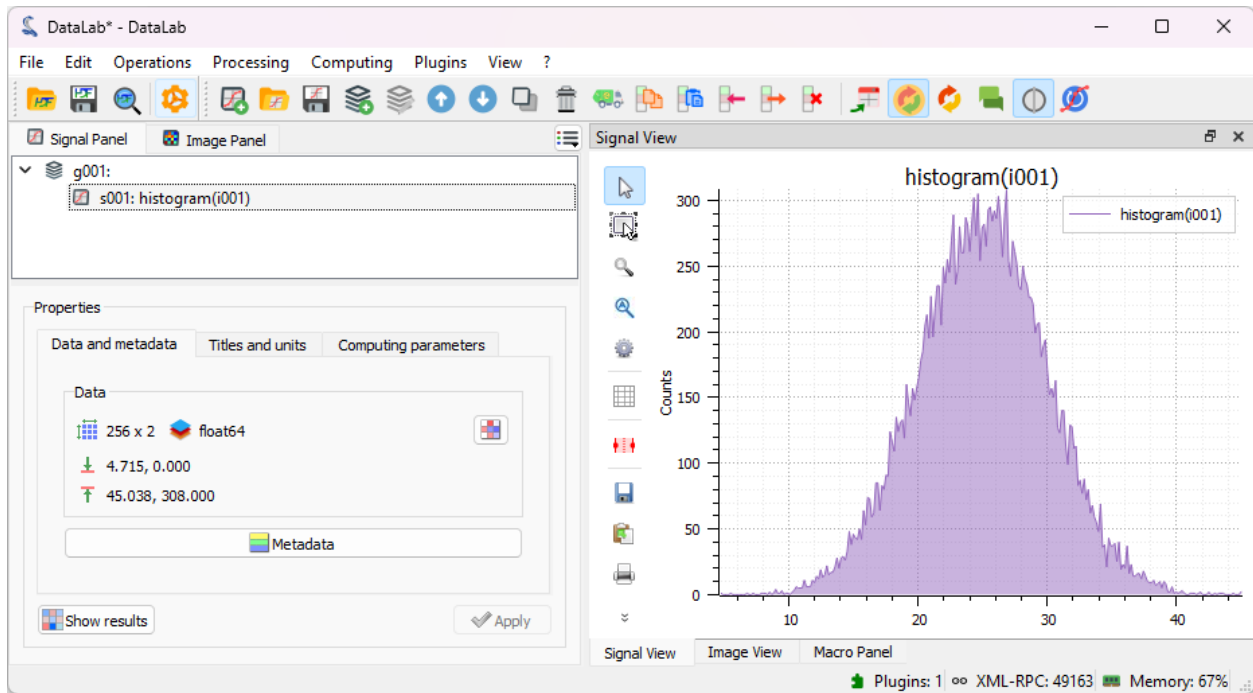


FIG. 43 – Exemple d'histogramme.

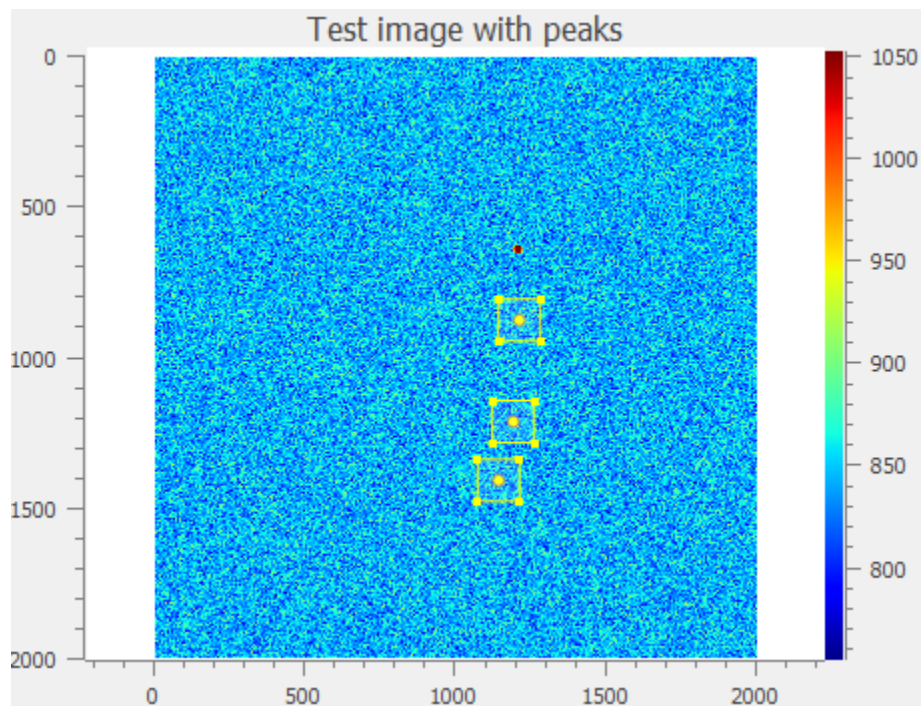


FIG. 44 – Exemple de détection de pics 2D.

## Détection de contours

Détecte automatiquement les contours et ajuste ces derniers par des cercles ou des ellipses, ou les représente directement par des polygones.

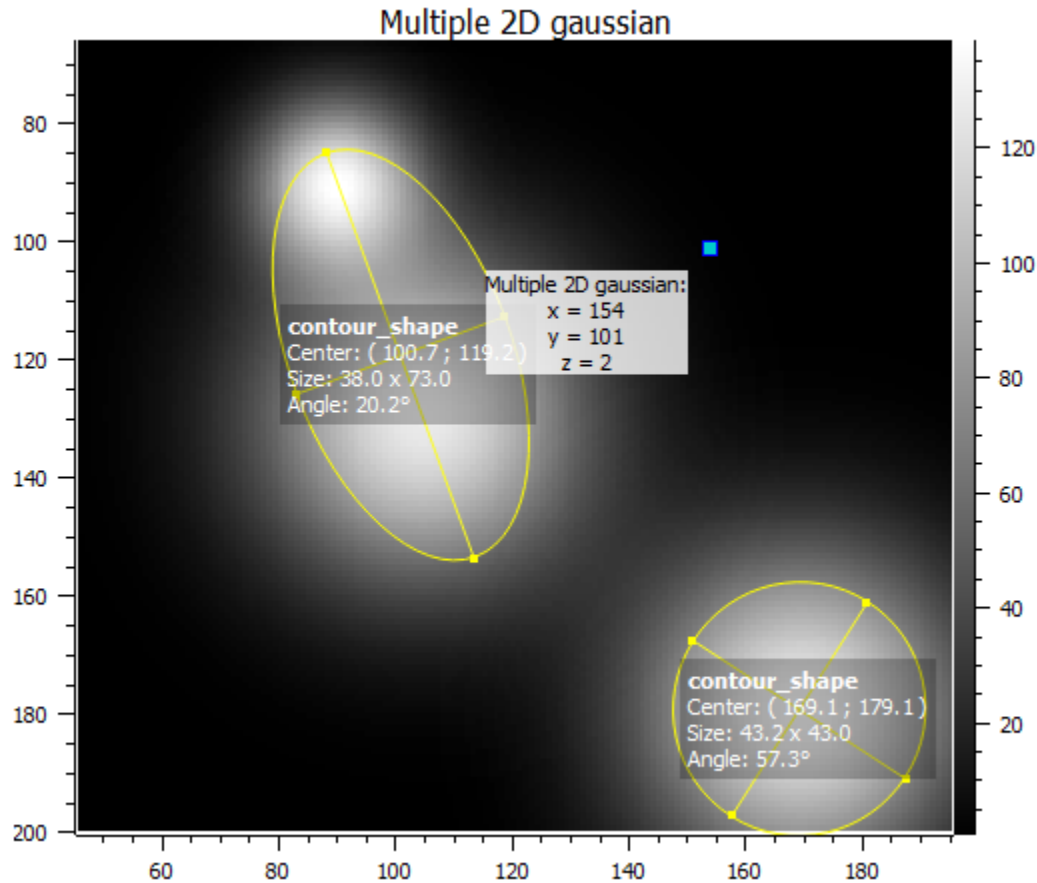


FIG. 45 – Exemple de détection de contours.

### Voir aussi :

Voir *Détection de contours* pour plus de détails sur l'algorithme et les paramètres associés.

**Note :** Les résultats de calcul scalaires sont systématiquement stockés dans les métadonnées. Les métadonnées sont attachées à l'image et sérialisées avec cette dernière par exemple lors de l'export d'une session de DataLab vers un fichier HDF5.

## Transformée de Hough circulaire

Détection de formes circulaires à partir d'une transformée de Hough (implémentation basée sur `skimage.transform.hough_circle_peaks`)

## Détection de taches

### Détection de taches (DOG)

Détection de taches basée sur la méthode de différence de gaussienne (DOG) (implémentation basée sur `skimage.feature.blob_dog`).

### Détection de taches (hessien)

Détection de taches basée sur la méthode du discriminant hessien (implémentation basée sur `skimage.feature.blob_doh`).

### Détection de taches (LOG)

Détection de taches basée sur la méthode du laplacien de gaussienne (LOG) (implémentation basée sur `skimage.feature.blob_log`).

### Détection de taches (OpenCV)

Détection de taches basée sur l'implémentation OpenCV de `SimpleBlobDetector`.

## Afficher les résultats

Affiche les résultats de toutes les analyses effectuées sur les images sélectionnées. Cela affiche le même tableau que celui affiché après avoir effectué un calcul.

## Tracer les résultats

Trace les résultats des analyses effectuées sur les images sélectionnées, avec des axes X et Y définis par l'utilisateur (p.ex. trace le rayon du cercle de contour en fonction du numéro de l'image).

## 2.4.6 Options d'affichage pour les images

Lorsque le « Panneau Image » est sélectionné, les menus et barres d'outils sont mis à jour pour fournir les actions liées aux images.

Le menu « Affichage » permet de visualiser l'image courante ou un groupe d'images. Il permet également d'afficher/cacher les titres, d'afficher/cacher le panneau de contraste, de rafraîchir la visualisation, etc.

## Voir dans une nouvelle fenêtre

Ouvre une nouvelle fenêtre pour visualiser les images sélectionnées.

Cette option permet d'afficher les images sélectionnées dans une nouvelle fenêtre, dans laquelle vous pouvez visualiser les données plus confortablement (par exemple, en maximisant la fenêtre) et ajouter ou modifier des annotations.

Lorsque vous cliquez sur le bouton « Annotations » dans la barre d'outils de la nouvelle fenêtre, le mode d'édition des annotations est activé (voir la section « Modifier les annotations » ci-dessous).

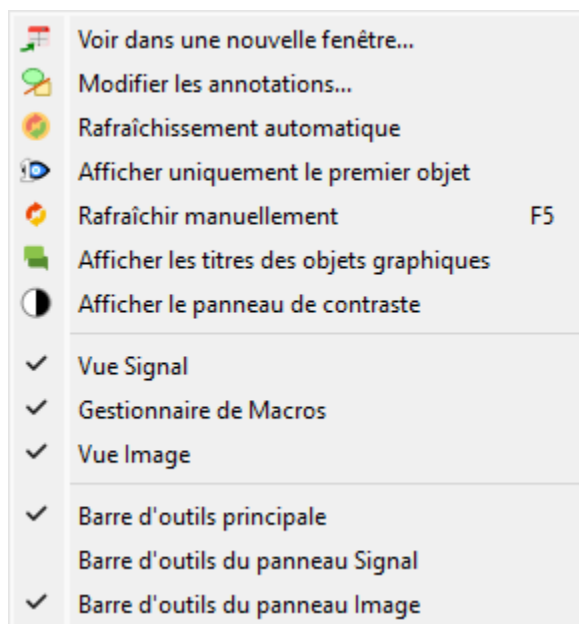


FIG. 46 – Capture d’écran du menu « Affichage ».

## Modifier les annotations

Ouvre une nouvelle fenêtre pour modifier les annotations.

Cette option vous permet de modifier les annotations des images sélectionnées dans une fenêtre séparée. Cela équivaut à sélectionner l’option « Voir dans une nouvelle fenêtre » puis à cliquer sur le bouton « Annotations » dans la barre d’outils de la nouvelle fenêtre.

Les annotations sont utilisées pour ajouter du texte, des lignes, des rectangles, des ellipses et d’autres formes géométriques aux images. Elles sont utiles pour mettre en évidence des régions d’intérêt, ajouter des commentaires, marquer des points, etc.

---

**Note :** Les annotations sont enregistrées dans les métadonnées de l’image, elles sont donc persistantes et seront affichées chaque fois que vous visualiserez l’image.

---

Le mode opératoire typique pour modifier les annotations est le suivant :

1. Sélectionnez l’option « Modifier les annotations ».
2. Dans la nouvelle fenêtre, ajoutez des annotations en cliquant sur les boutons correspondants en bas de la fenêtre.
3. Personnalisez les annotations en modifiant leurs propriétés (par exemple, le texte, la couleur, la position, etc.) en utilisant l’option « Paramètres » dans le menu contextuel des annotations.
4. Quand vous avez terminé, cliquez sur le bouton « OK » pour enregistrer les annotations. Cela fermera la fenêtre et les annotations seront enregistrées dans les métadonnées de l’image et seront affichées dans la fenêtre principale.

---

**Note :** Les annotations peuvent être copiées d’une image à une autre en utilisant les fonctionnalités « copier/coller les métadonnées ».

---

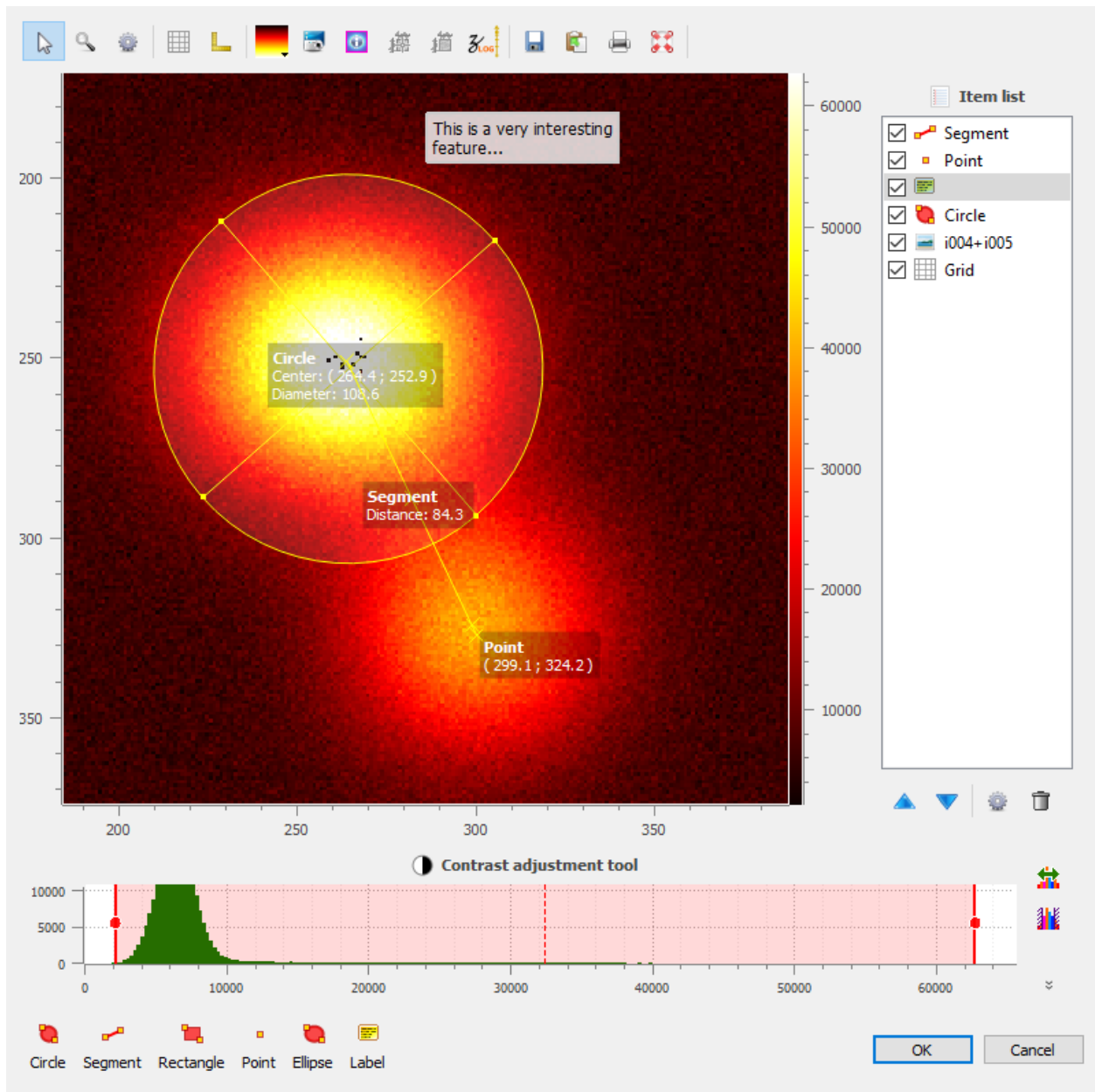


FIG. 47 – Les annotations peuvent être ajoutées dans la vue séparée.

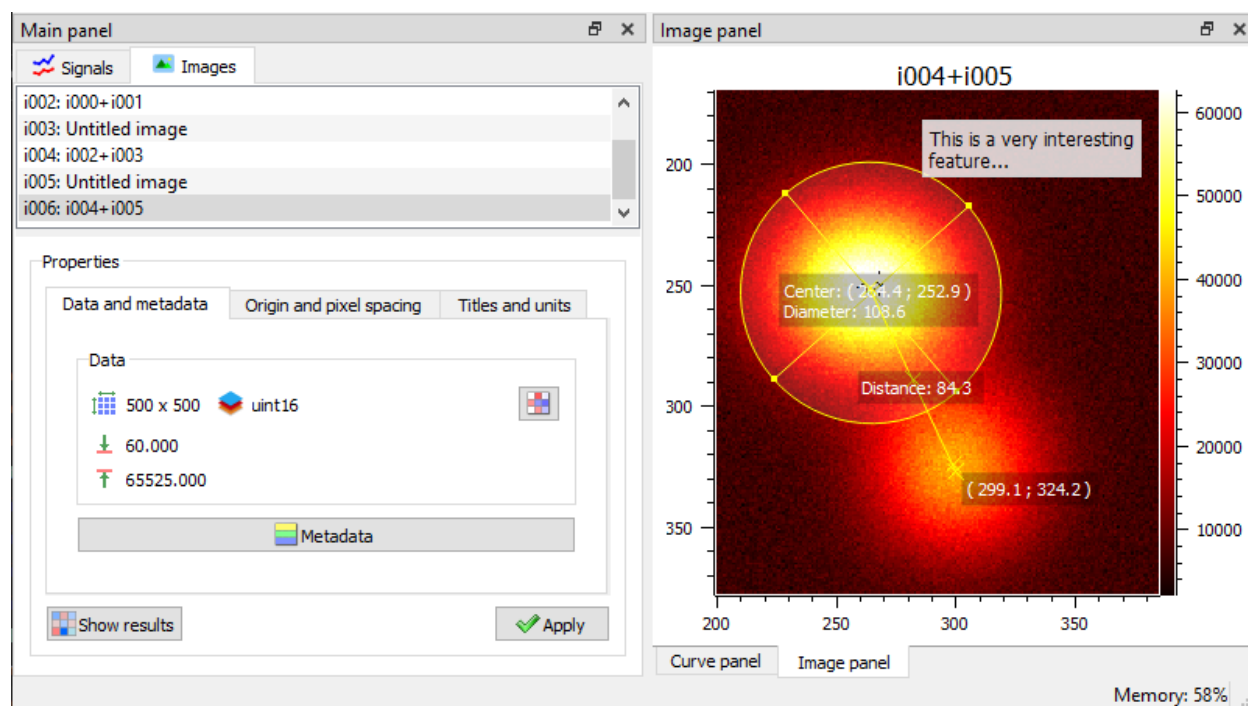


FIG. 48 – Les annotations font désormais partie des métadonnées de l'image.

## Rafraîchissement automatique

Rafraîchit automatiquement la visualisation quand les données changent :

- Si activé (par défaut), la visualisation est automatiquement rafraîchie quand les données changent.
- Si désactivé, la visualisation n'est pas rafraîchie tant que vous ne l'avez pas rafraîchie manuellement en utilisant l'entrée de menu « Rafraîchir manuellement ».

Même si l'algorithme de rafraîchissement est optimisé, il peut prendre du temps pour rafraîchir la visualisation quand les données changent, surtout quand le jeu de données est grand. Par conséquent, vous pouvez désactiver le rafraîchissement automatique quand vous travaillez avec des données volumineuses, et l'activer à nouveau quand vous avez terminé. Cela évitera des rafraîchissements inutiles.

## Rafraîchir manuellement

Rafraîchir la visualisation manuellement.

Cela déclenche un rafraîchissement de la visualisation. C'est utile quand le rafraîchissement automatique est désactivé, ou quand vous voulez forcer un rafraîchissement de la visualisation.

### Afficher le panneau de contraste

Affiche/cache le panneau de réglage du contraste.

### Afficher les titres des objets graphiques

Affiche/cache les titres des objets graphiques liés aux résultats d'analyse et aux annotations.

Cette option vous permet d'afficher ou de masquer les titres des objets graphiques (par exemple, les titres des résultats d'analyse ou des annotations). Masquer les titres peut être utile lorsque vous voulez visualiser les données sans être perturbé, ou s'il y a trop de titres et qu'ils se chevauchent.

## 2.4.7 Détection de pics 2D

DataLab fournit une fonctionnalité de « Détection de pics 2D » qui est basée sur un algorithme de filtrage minimum-maximum.

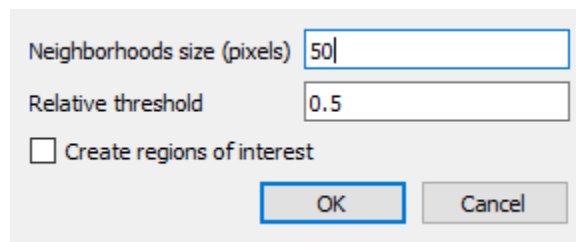


FIG. 49 – Paramètres de détection de pics 2D.

#### La fonctionnalité s'utilise de la façon suivante :

- Créer ou ouvrir une image dans l'espace de travail de DataLab
- Sélectionner « Détection de pics 2D » dans le menu Analyse »
- Saisir les paramètres « Taille de voisinage » et « Seuil relatif »
- Cocher « Créer des régions d'intérêt » si vous souhaitez que des ROI soient définies pour chaque pic détecté (ce qui peut s'avérer utile en cas de calcul ultérieur sur chaque pic détecté, tel que par exemple une détection de contour).

#### Les résultats sont affichés dans un tableau :

- Chaque ligne est associée à un pic détecté
- La première colonne contient l'indice de la ROI (0 si aucune ROI n'est définie)
- Les deuxième et troisième colonnes contiennent les coordonnées des pics

#### L'algorithme de détection de pics 2D fonctionne de la manière suivante :

- Tout d'abord, des images filtrées minimum-maximum sont calculées en utilisant un algorithme de fenêtre glissante dont la taille est définie par l'utilisateur (implémentation basée sur `scipy.ndimage.minimum_filter` et `scipy.ndimage.maximum_filter`)
- Ensuite, la différence entre ces deux images filtrées est échantillonnée à une valeur correspondant à un seuil défini par l'utilisateur
- L'image résultante est étiquetée en utilisant `scipy.ndimage.label`
- Les coordonnées des pics sont ensuite obtenues en calculant le centre de chaque étiquette
- Les doublons sont éventuellement supprimés

#### Les paramètres de la détection de pics 2D sont les suivants :

- « Taille de voisinage » : taille de la fenêtre glissante (cf. plus haut)

Results - NumPy array (read only)			
	ROI	x	y
Peaks(i000)	0	1366	638
Peaks(i000)	0	1416	1069
Peaks(i000)	0	1018	1135
Peaks(i000)	0	828	1229

Format    Resize    ☒ Background color

Close

FIG. 50 – Résultats d'une détection de pics 2D (voir le test « peak2d\_app.py »)

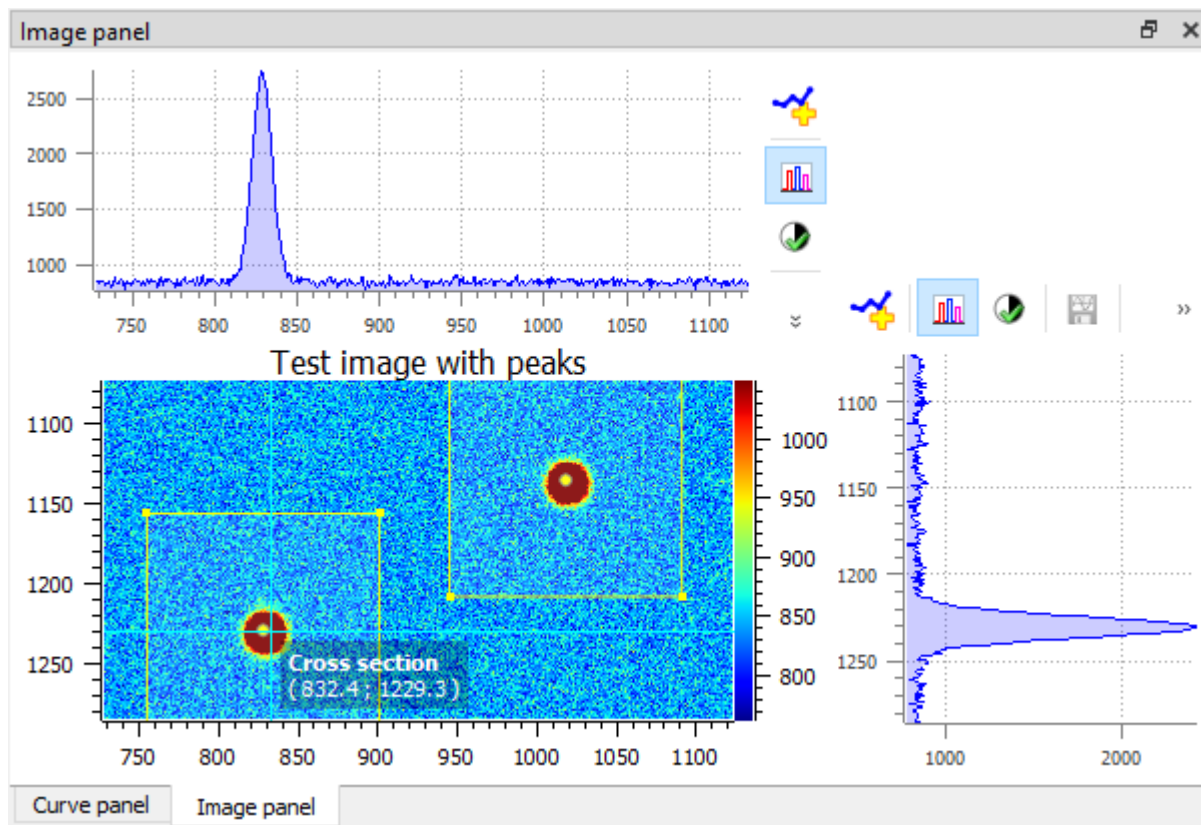


FIG. 51 – Exemple de détection de pics 2D.

— « Seuil relatif » : seuil de détection

La fonctionnalité est basée sur la fonction `get_2d_peaks_coords` du module `cdl.algorithms` :

```
def get_2d_peaks_coords(
    data: np.ndarray, size: int | None = None, level: float = 0.5
) -> np.ndarray:
    """Detect peaks in image data, return coordinates.

    If neighborhoods size is None, default value is the highest value
    between 50 pixels and the 1/40th of the smallest image dimension.

    Detection threshold level is relative to difference
    between data maximum and minimum values.

    Args:
        data: Input data
        size: Neighborhood size (default: None)
        level: Relative level (default: 0.5)

    Returns:
        Coordinates of peaks
    """
    if size is None:
        size = max(min(data.shape) // 40, 50)
    data_max = spi.maximum_filter(data, size)
    data_min = spi.minimum_filter(data, size)
    data_diff = data_max - data_min
    diff = (data_max - data_min) > get_absolute_level(data_diff, level)
    maxima = data == data_max
    maxima[diff == 0] = 0
    labeled, _num_objects = spi.label(maxima)
    slices = spi.find_objects(labeled)
    coords = []
    for dy, dx in slices:
        x_center = int(0.5 * (dx.start + dx.stop - 1))
        y_center = int(0.5 * (dy.start + dy.stop - 1))
        coords.append((x_center, y_center))
    if len(coords) > 1:
        # Eventually removing duplicates
        dist = distance_matrix(coords)
        for index in reversed(np.unique(np.where((dist < size) & (dist > 0))[1])):
            coords.pop(index)
    return np.array(coords)
```

## 2.4.8 Détection de contours

DataLab fournit une fonctionnalité de « Détection de contours » qui est basée sur l'algorithme des [marching cubes](#)

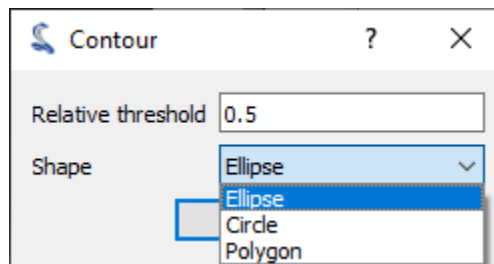


FIG. 52 – Paramètres de détection de contours.

**La fonctionnalité s'utilise de la façon suivante :**

- Créer ou ouvrir une image dans l'espace de travail de DataLab
- Créer éventuellement une ROI autour de la zone cible de l'image
- Sélectionner « Détection de contours » dans le menu Analyse »
- Saisir le paramètre « Forme » (« Ellipse », « Cercle » ou « Polygone »)

	ROI	x0	y0	x1	y1	x2
i001: contour_shape	0	110	150.125	109	150.375	108
i001: contour_shape	0	160.75	199	160	198.625	159

FIG. 53 – Résultats de la détection de contours (cf. test « contour\_app.py »)

**Les résultats sont affichés dans un tableau :**

- Chaque ligne est associée à un contour
- La première colonne contient l'indice de la ROI (0 si aucune ROI n'est définie)
- Les colonnes suivantes présentent les coordonnées des contours : 4 colonnes pour les cercles (coordonnées du diamètre) et 8 colonnes pour les ellipses (coordonnées des diamètres)

**L'algorithme de détection de contours fonctionne de la manière suivante :**

- Tout d'abord, les isocontours sont calculés (l'implémentation est basée sur [ski-image.measure.find\\_contours.find\\_contours](#))
- Ensuite, chaque contour est ajusté à une ellipse (ou à un cercle)

La fonctionnalité est basée sur la fonction `get_contour_shapes` du module `cdl.computation` :

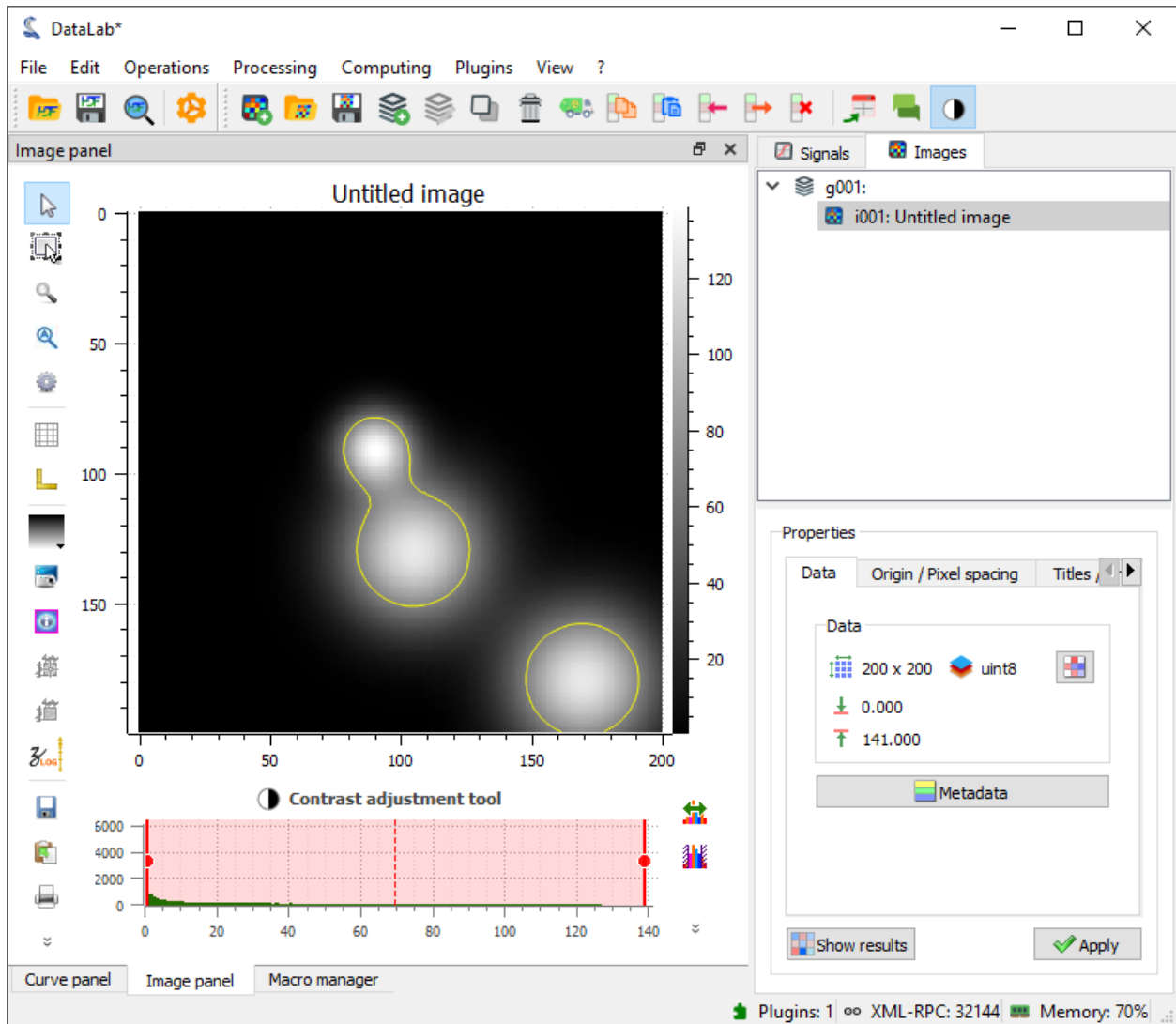


FIG. 54 – Exemple de détection de contours.

```

def get_contour_shapes(
    data: np.ndarray | ma.MaskedArray,
    shape: Literal["circle", "ellipse", "polygon"] = "ellipse",
    level: float = 0.5,
) -> np.ndarray:
    """Find iso-valued contours in a 2D array, above relative level (.5 means
    ↪FWHM),
    then fit contours with shape ('ellipse' or 'circle')

    Args:
        data: Input data
        shape: Shape to fit. Default is 'ellipse'
        level: Relative level (default: 0.5)

    Returns:
        Coordinates of shapes
    """
    # pylint: disable=too-many-locals
    assert shape in ("circle", "ellipse", "polygon")
    contours = measure.find_contours(data, level=get_absolute_level(data,
    ↪level))
    coords = []
    for contour in contours:
        # `contour` is a (N, 2) array (rows, cols): we need to check if all
    ↪those
        # coordinates are masked: if so, we skip this contour
        if isinstance(data, ma.MaskedArray) and np.all(
            data.mask[contour[:, 0].astype(int), contour[:, 1].astype(int)]
        ):
            continue
        if shape == "circle":
            model = measure.CircleModel()
            if model.estimate(contour):
                yc, xc, r = model.params
                if r <= 1.0:
                    continue
                coords.append([xc, yc, r])
        elif shape == "ellipse":
            model = measure.EllipseModel()
            if model.estimate(contour):
                yc, xc, b, a, theta = model.params
                if a <= 1.0 or b <= 1.0:
                    continue
                coords.append([xc, yc, a, b, theta])
        elif shape == "polygon":
            # `contour` is a (N, 2) array (rows, cols): we need to convert it
            # to a list of x, y coordinates flattened in a single list
            coords.append(contour[:, :-1].flatten())
        else:
            raise NotImplementedError(f"Invalid contour model {model}")
    if shape == "polygon":
        # `coords` is a list of arrays of shape (N, 2) where N is the number of
    ↪points

```

(suite sur la page suivante)

(suite de la page précédente)

```
# that can vary from one array to another, so we need to padd with  
↪NaNs each  
# array to get a regular array:  
max_len = max(coord.shape[0] for coord in coords)  
arr = np.full((len(coords), max_len), np.nan)  
for i_row, coord in enumerate(coords):  
    arr[i_row, : coord.shape[0]] = coord  
return arr  
return np.array(coords)
```



L'interface de programmation d'application (API) publique de DataLab offre un ensemble de fonctions pour accéder aux fonctionnalités de DataLab. Ces fonctions sont disponibles dans divers sous-modules du package *cdl*. Le tableau suivant répertorie les sous-modules disponibles et leurs contenus respectifs :

Sous-module	Contenu
<i>cdl.algorithms</i>	Algorithmes pour l'analyse de données, qui opèrent sur des tableaux NumPy
<i>cdl.param</i>	Module de commodité pour accéder aux ensembles de paramètres DataLab (instances d'objets <i>guidata.dataset.DataSet</i> )
<i>cdl.obj</i>	Module de commodité pour accéder aux objets DataLab ( <i>cdl.obj.SignalObj</i> ou <i>cdl.obj.ImageObj</i> ) et fonctions associées
<i>cdl.computation</i>	Fonctions de calcul, qui opèrent sur les objets DataLab ( <i>cdl.obj.SignalObj</i> ou <i>cdl.obj.ImageObj</i> )
<i>cdl.proxy</i>	Objets proxy pour accéder à l'interface DataLab à partir d'un script Python ou d'une application distante

## 3.1 Algorithmes (*cdl.algorithms*)

Ce paquet contient les algorithmes utilisés par le projet DataLab. Ces algorithmes opèrent directement sur des tableaux NumPy et sont conçus pour être utilisés dans le pipeline DataLab, mais peuvent également être utilisés indépendamment.

**Voir aussi :**

Le module *cdl.algorithms* est le point d'entrée principal pour les algorithmes DataLab lors de la manipulation de tableaux NumPy. Voir le module *cdl.computation* pour les algorithmes qui opèrent directement sur les objets DataLab (c'est-à-dire *cdl.obj.SignalObj* et *cdl.obj.ImageObj*).

Les algorithmes sont organisés en sous-paquets en fonction de leur objectif. Les sous-paquets suivants sont disponibles :

- *cdl.algorithms.signal* : Algorithmes de traitement du signal

- `cdl.algorithms.image` : Algorithmes de traitement d'image
- `cdl.algorithms.datatypes` : Algorithmes de conversion de type de données
- `cdl.algorithms.coordinates` : Algorithmes de conversion de coordonnées

### 3.1.1 Algorithmes de traitement du signal

`cdl.algorithms.signal.normalize(yin : ndarray, parameter : Literal['maximum', 'amplitude', 'area', 'energy', 'rms'] = 'maximum') → ndarray`

Normaliser le tableau d'entrée à un paramètre donné.

**Paramètres**

- **yin** – Tableau d'entrée
- **parameter** – Paramètre de normalisation. Par défaut : « maximum »

**Renvoie**

Tableau normalisé

`cdl.algorithms.signal.fft1d(x : ndarray, y : ndarray, shift : bool = True) → tuple[ndarray, ndarray]`

Calculer la FFT sur les données X,Y.

**Paramètres**

- **x** – Données X
- **y** – Données Y
- **shift** – Décaler la fréquence nulle vers le centre du spectre. Par défaut : True.

**Renvoie**

Données X, Y (tuple)

`cdl.algorithms.signal.ifft1d(x : ndarray, y : ndarray, shift : bool = True) → tuple[ndarray, ndarray]`

Calculer la iFFT sur les données X,Y.

**Paramètres**

- **x** – Données X
- **y** – Données Y
- **shift** – Décaler la fréquence nulle vers le centre du spectre. Par défaut : True.

**Renvoie**

Données X, Y (tuple)

`cdl.algorithms.signal.magnitude_spectrum(x : ndarray, y : ndarray, log_scale : bool = False) → tuple[ndarray, ndarray]`

Calculer le spectre de magnitude.

**Paramètres**

- **x** – Données X
- **y** – Données Y
- **log\_scale** – Utiliser une échelle logarithmique. Par défaut : False.

**Renvoie**

Spectre de magnitude (données X, Y)

`cdl.algorithms.signal.phase_spectrum(x : ndarray, y : ndarray) → tuple[ndarray, ndarray]`

Calculer le spectre de phase.

**Paramètres**

- **x** – Données X
- **y** – Données Y

**Renvoie**

Spectre de phase en degrés (données X, Y)

`cdl.algorithms.signal.psd(x : ndarray, y : ndarray, log_scale : bool = False) → tuple[ndarray, ndarray]`

Calculer la densité spectrale de puissance (PSD), en utilisant la méthode de Welch.

**Paramètres**

- **x** – Données X
- **y** – Données Y
- **log\_scale** – Utiliser une échelle logarithmique. Par défaut : False.

**Renvoie**

Données X, Y (tuple)

**Type renvoyé**

Power Spectral Density (PSD)

`cdl.algorithms.signal.sort_frequencies(x : ndarray, y : ndarray) → ndarray`

Trier à partir des données X,Y en calculant FFT(y).

**Paramètres**

- **x** – Données X
- **y** – Données Y

**Renvoie**

Fréquences triées par ordre croissant

`cdl.algorithms.signal.peak_indices(y, thres : float = 0.3, min_dist : int = 1, thres_abs : bool = False) → ndarray`

Routine de détection de pics.

Recherche l'index numérique des pics dans y en prenant sa première différence d'ordre. En utilisant les paramètres *thres* et *min\_dist*, il est possible de réduire le nombre de pics détectés. y doit être signé.

**Paramètres**

- **y** (*ndarray (signed)*) – Données d'amplitude 1D à rechercher pour les pics.
- **thres** (*float between [0., 1.]*) – Seuil normalisé. Seuls les pics dont l'amplitude est supérieure au seuil seront détectés.
- **min\_dist** (*int*) – Distance minimale entre chaque pic détecté. Le pic avec l'amplitude la plus élevée est préféré pour satisfaire cette contrainte.
- **thres\_abs** (*boolean*) – Si True, la valeur *thres* sera interprétée comme une valeur absolue, au lieu d'un seuil normalisé.

**Renvoie**

Tableau contenant les index numériques des pics qui ont été détectés

**Type renvoyé**

ndarray

`cdl.algorithms.signal.xpeak(x : ndarray, y : ndarray) → float`

Retourne la position X du pic par défaut (en supposant un seul pic).

**Paramètres**

- **x** – Données X
- **y** – Données Y

**Renvoie**

Position X du pic

`cdl.algorithms.signal.interpolate(x : ndarray, y : ndarray, xnew : ndarray, method : Literal['linear', 'spline', 'quadratic', 'cubic', 'barycentric', 'pchip'], fill_value : float | None = None) → ndarray`

Interpoler les données.

**Paramètres**

- **x** – Données X
- **y** – Données Y

- **xnew** – Nouvelles données X
- **method** – Méthode d’interpolation
- **fill\_value** – Valeur de remplissage. Par défaut : None. Cette valeur est utilisée pour remplir les points demandés en dehors de la plage de données X. Elle n’est utilisée que si l’argument de méthode est “linear”, “cubic” ou “pchip”.

**Renvois**

Données Y interpolées

```
cdl.algorithms.signal.windowing(y : ndarray, method : Literal['barthann', 'bartlett', 'blackman',  
                        'blackman-harris', 'bohman', 'boxcar', 'cosine', 'exponential', 'flat-top',  
                        'hamming', 'hanning', 'lanczos', 'nutall', 'parzen', 'rectangular', 'taylor',  
                        'tukey', 'kaiser', 'gaussian'] = 'hamming', alpha : float = 0.5, beta : float =  
                        14.0, sigma : float = 7.0) → ndarray
```

Appliquer une fenêtre aux données d’entrée.

**Paramètres**

- **x** – Données X
- **y** – Données Y
- **method** – Fonction de fenêtrage. Par défaut : « hamming ».
- **alpha** – Paramètre de la fenêtre de Tukey. Par défaut : 0.5.
- **beta** – Paramètre de la fenêtre de Kaiser. Par défaut : 14.0.
- **sigma** – Paramètre de la fenêtre gaussienne. Par défaut : 7.0.

**Renvois**

Données Y fenêtrées

```
class cdl.algorithms.signal.FitModel
```

Classe de base du modèle de régression

```
abstract classmethod func(x, amp, sigma, x0, y0)
```

Retourne la fonction de régression

```
classmethod get_amp_from_amplitude(amplitude, sigma)
```

Retourne amp à partir de l’amplitude de la fonction et de sigma

```
classmethod amplitude(amp, sigma)
```

Retourne l’amplitude de la fonction

```
abstract classmethod fwhm(amp, sigma)
```

Retourne la LMH de la fonction

```
classmethod half_max_segment(amp, sigma, x0, y0)
```

Retourne les coordonnées du segment pour l’intersection y=half-maximum

```
class cdl.algorithms.signal.GaussianModel
```

Modèle de régression gaussien à 1 dimension

```
classmethod func(x, amp, sigma, x0, y0)
```

Retourne la fonction de régression

```
classmethod get_amp_from_amplitude(amplitude, sigma)
```

Retourne amp à partir de l’amplitude de la fonction et de sigma

```
classmethod amplitude(amp, sigma)
```

Retourne l’amplitude de la fonction

```
classmethod fwhm(amp, sigma)
```

Retourne la LMH de la fonction

**class** `cdl.algorithms.signal.LorentzianModel`

Modèle de régression lorentzien à 1 dimension

**classmethod** `func(x, amp, sigma, x0, y0)`

Retourne la fonction de régression

**classmethod** `get_amp_from_amplitude(amplitude, sigma)`

Retourne amp à partir de l'amplitude de la fonction et de sigma

**classmethod** `amplitude(amp, sigma)`

Retourne l'amplitude de la fonction

**classmethod** `fwhm(amp, sigma)`

Retourne la LMH de la fonction

**class** `cdl.algorithms.signal.VoigtModel`

Modèle de régression Voigt à 1 dimension

**classmethod** `func(x, amp, sigma, x0, y0)`

Retourne la fonction de régression

**classmethod** `fwhm(amp, sigma)`

Retourne la LMH de la fonction

`cdl.algorithms.signal.find_nearest_zero_point_idx(y : ndarray) → ndarray`

Rechercher les indices x où le y correspondant est le plus proche de zéro

**Paramètres**

**y** – Données Y

**Renvoie**

Indices des points juste avant ou au croisement à zéro

`cdl.algorithms.signal.find_x_at_value(x : ndarray, y : ndarray, value : float) → ndarray`

Rechercher la valeur x où la valeur y est la plus proche de la valeur donnée en utilisant une interpolation linéaire pour déduire la valeur x précise.

**Paramètres**

— **x** – Données X

— **y** – Données Y

— **value** – Valeur à rechercher

**Renvoie**

An array of x values where the y value is the closest to the given value (empty array if no zero crossing is found)

`cdl.algorithms.signal.bandwidth(data : ndarray, level : float = 3.0) → tuple[float, float, float, float]`

Calculer la largeur de bande du signal à un niveau donné.

**Paramètres**

— **data** – Données X,Y

— **level** – Niveau en dB auquel la largeur de bande est calculée. Par défaut : 3.0.

**Renvoie**

coordonnées du segment

**Type renvoyé**

Bandwidth of the signal at the given level

`cdl.algorithms.signal.contrast(y : ndarray) → float`

Calculer le contraste

**Paramètres**

**y** – Tableau d'entrée

**Renvois**

Contraste

`cdl.algorithms.signal.sinusoidal_model(x : ndarray, a : float, f : float, phi : float, offset : float) → ndarray`

Fonction de modèle sinusoïdal.

`cdl.algorithms.signal.sinusoidal_fit(x : ndarray, y : ndarray) → tuple[tuple[float, float, float, float], float]`

Adapter un modèle sinusoïdal aux données d'entrée.

**Paramètres**

- **x** – Données X
- **y** – Données Y

**Renvois**

Un tuple contenant les paramètres d'ajustement (amplitude, fréquence, phase, décalage) et les résidus

`cdl.algorithms.signal.sinus_frequency(x : ndarray, y : ndarray) → float`

Calculer la fréquence d'un signal sinusoïdal.

**Paramètres**

- **x** – données x du signal
- **y** – données y du signal

**Renvois**

Fréquence du signal sinusoïdal

`cdl.algorithms.signal.enob(x : ndarray, y : ndarray, full_scale : float = 1.0) → float`

Calculer le nombre effectif de bits (ENOB).

**Paramètres**

- **x** – données x du signal
- **y** – données y du signal
- **full\_scale** – Pleine échelle (V). Par défaut : 1.0.

**Renvois**

Nombre effectif de bits (ENOB)

`cdl.algorithms.signal.sinad(x : ndarray, y : ndarray, full_scale : float = 1.0, unit : Literal['dBc', 'dBFS'] = 'dBc') → float`

Calculer le rapport signal/bruit et distorsion (SINAD).

**Paramètres**

- **x** – données x du signal
- **y** – données y du signal
- **full\_scale** – Pleine échelle (V). Par défaut : 1.0.
- **unit** – Unité des données d'entrée. Les valeurs valides sont "dBc" et "dBFS". Par défaut : "dBc".

**Renvois**

Rapport signal/bruit et distorsion (SINAD)

`cdl.algorithms.signal.thd(x : ndarray, y : ndarray, full_scale : float = 1.0, unit : Literal['dBc', 'dBFS'] = 'dBc', nb_harm : int = 5) → float`

Calculer la distorsion harmonique totale (THD).

**Paramètres**

- **x** – données x du signal
- **y** – données y du signal
- **full\_scale** – Pleine échelle (V). Par défaut : 1.0.
- **unit** – Unité des données d'entrée. Les valeurs valides sont "dBc" et "dBFS". Par défaut : "dBc".

— **nb\_harm** – Nombre d’harmoniques à considérer. Par défaut : 5.

**Renvoie**

Distorsion harmonique totale (THD)

```
cdl.algorithms.signal.sfdr(x : ndarray, y : ndarray, full_scale : float = 1.0, unit : Literal['dBc', 'dBFS'] = 'dBc') → float
```

Calculer la plage dynamique sans spurious (SFDR).

**Paramètres**

- **x** – données x du signal
- **y** – données y du signal
- **full\_scale** – Pleine échelle (V). Par défaut : 1.0.
- **unit** – Unité des données d’entrée. Les valeurs valides sont “dBc” et “dBFS”. Par défaut : “dBc”.

**Renvoie**

Plage dynamique sans spurious (SFDR)

```
cdl.algorithms.signal.snr(x : ndarray, y : ndarray, full_scale : float = 1.0, unit : Literal['dBc', 'dBFS'] = 'dBc') → float
```

Calculer le rapport signal/bruit (SNR).

**Paramètres**

- **x** – données x du signal
- **y** – données y du signal
- **full\_scale** – Pleine échelle (V). Par défaut : 1.0.
- **unit** – Unité des données d’entrée. Les valeurs valides sont “dBc” et “dBFS”. Par défaut : “dBc”.

**Renvoie**

Rapport signal/bruit (SNR)

```
cdl.algorithms.signal.sampling_period(x : ndarray) → float
```

Calculer la période d’échantillonnage

**Paramètres**

- x** – Données X

**Renvoie**

Période d’échantillonnage

```
cdl.algorithms.signal.sampling_rate(x : ndarray) → float
```

Calculer le taux d’échantillonnage moyen

**Paramètres**

- x** – Données X

**Renvoie**

Taux d’échantillonnage

```
cdl.algorithms.signal.fwhm(data : ndarray, method : Literal['zero-crossing', 'gauss', 'lorentz', 'voigt'] = 'zero-crossing', xmin : float | None = None, xmax : float | None = None) → tuple[float, float, float, float]
```

Calculer la largeur totale à mi-hauteur (FWHM) des données d’entrée

**Paramètres**

- **data** – Données X,Y
- **method** – Méthode de calcul. Deux types de méthodes sont pris en charge : une méthode de croisement à zéro et des méthodes d’ajustement (basées sur divers modèles : Gauss, Lorentz, Voigt). Par défaut : « zero-crossing ».
- **xmin** – Borne X inférieure pour l’ajustement. Par défaut : None (pas de borne inférieure, c’est-à-dire que l’ajustement commence à partir du premier point).

- **xmax** – Borne X supérieure pour l’ajustement. Par défaut : None (pas de borne supérieure, c’est-à-dire que l’ajustement se termine au dernier point)

**Renvoie**

Coordonnées du segment LMH

`cdl.algorithms.signal.fw1e2(data : ndarray) → tuple[float, float, float, float]`

Calculer la largeur totale à  $1/e^2$  des données d’entrée (en utilisant un ajustement de modèle gaussien).

**Paramètres**

**data** – Données X,Y

**Renvoie**

Coordonnées du segment LMH à  $1/e^2$

### 3.1.2 Algorithmes de traitement d’image

`cdl.algorithms.image.scale_data_to_min_max(data : ndarray, zmin : float | int, zmax : float | int) → ndarray`

Mettre à l’échelle le tableau *data* pour s’adapter à la plage dynamique [zmin, zmax]

**Paramètres**

- **data** – Données d’entrée
- **zmin** – Valeur minimale des données de sortie
- **zmax** – Valeur maximale des données de sortie

**Renvoie**

Données mises à l’échelle

`cdl.algorithms.image.normalize(data : ndarray, parameter : Literal['maximum', 'amplitude', 'area', 'energy', 'rms'] = 'maximum') → ndarray`

Normaliser le tableau d’entrée à un paramètre donné.

**Paramètres**

- **data** – Données d’entrée
- **parameter** – Paramètre de normalisation (par défaut : « maximum »)

**Renvoie**

Tableau normalisé

`cdl.algorithms.image.fft2d(z : ndarray, shift : bool = True) → ndarray`

Calculer la FFT du tableau complexe *z*

**Paramètres**

- **z** – Données d’entrée
- **shift** – Décaler la fréquence nulle vers le centre (par défaut : True)

**Renvoie**

FFT des données d’entrée

`cdl.algorithms.image.ifft2d(z : ndarray, shift : bool = True) → ndarray`

Calculer la FFT inverse du tableau complexe *z*

**Paramètres**

- **z** – Données d’entrée
- **shift** – Décaler la fréquence nulle vers le centre (par défaut : True)

**Renvoie**

FFT inverse des données d’entrée

`cdl.algorithms.image.magnitude_spectrum(z : ndarray, log_scale : bool = False) → ndarray`

Calculer le spectre de magnitude du tableau complexe *z*

**Paramètres**

- **z** – Données d’entrée
- **log\_scale** – Utiliser une échelle logarithmique (par défaut : False)

**Renvoie**

Spectre de magnitude des données d’entrée

`cdl.algorithms.image.phase_spectrum(z : ndarray) → ndarray`

Calculer le spectre de phase du tableau complexe *z*

**Paramètres**

- z** – Données d’entrée

**Renvoie**

Spectre de phase des données d’entrée (en degrés)

`cdl.algorithms.image.psd(z : ndarray, log_scale : bool = False) → ndarray`

Calculer la densité spectrale de puissance du tableau complexe *z*

**Paramètres**

- **z** – Données d’entrée
- **log\_scale** – Utiliser une échelle logarithmique (par défaut : False)

**Renvoie**

Densité spectrale de puissance des données d’entrée

`cdl.algorithms.image.binning(data : ndarray, sx : int, sy : int, operation : Literal['sum', 'average', 'median', 'min', 'max'], dtype=None) → ndarray`

Effectuer le binning des pixels de l’image

**Paramètres**

- **data** – Données d’entrée
- **sx** – Binning le long de x (nombre de pixels à biner ensemble)
- **sy** – Binning le long de y (nombre de pixels à biner ensemble)
- **operation** – Opération de binning
- **dtype** – Type de données de sortie (par défaut : None, c’est-à-dire identique à l’entrée)

**Renvoie**

Données binnées

`cdl.algorithms.image.flatfield(rawdata : ndarray, flatdata : ndarray, threshold : float | None = None) → ndarray`

Calculer la correction du champ plat

**Paramètres**

- **rawdata** – Données brutes
- **flatdata** – Données de champ plat
- **threshold** – Seuil de correction du champ plat (par défaut : None)

**Renvoie**

Données corrigées du champ plat

`cdl.algorithms.image.get_centroid_fourier(data : ndarray) → tuple[float, float]`

Retourne le centre de gravité de l’image en utilisant l’algorithme de Fourier

**Paramètres**

- data** – Données d’entrée

**Renvoie**

Coordonnées du centre de gravité (ligne, colonne)

`cdl.algorithms.image.get_absolute_level(data : ndarray, level : float) → float`

Retourne le niveau absolu

**Paramètres**

- **data** – Données d’entrée

— **level** – Niveau relatif (0.0 à 1.0)

**Renvoie**

Niveau absolu

`cdl.algorithms.image.get_enclosing_circle(data : ndarray, level : float = 0.5) → tuple[int, int, float]`

Retourne (x, y, rayon) pour le contour du cercle contenant les valeurs d'image au-dessus du niveau relatif du seuil (.5 signifie FWHM)

**Paramètres**

- **data** – Données d'entrée
- **level** – Niveau relatif (par défaut : 0.5)

**Renvoie**

Un tuple (x, y, rayon)

**Lève**

**ValueError** – Aucun contour n'a été trouvé

`cdl.algorithms.image.get_radial_profile(data : ndarray, center : tuple[int, int]) → tuple[ndarray, ndarray]`

Retourne le profil radial des données d'image

**Paramètres**

- **data** – Tableau d'entrée (2D)
- **center** – Coordonnées du centre du profil (x, y)

**Renvoie**

Profil radial (X, Y) où X est la distance du centre (tableau 1D) et Y est la valeur moyenne des pixels à cette distance (tableau 1D)

`cdl.algorithms.image.distance_matrix(coords : list) → ndarray`

Retourne la matrice de distance à partir des coordonnées

**Paramètres**

- coords** – Liste des coordonnées

**Renvoie**

Matrice de distance

`cdl.algorithms.image.get_2d_peaks_coords(data : ndarray, size : int | None = None, level : float = 0.5) → ndarray`

Détecte les pics dans les données d'image, retourne les coordonnées.

Si la taille du voisinage est None, la valeur par défaut est la plus élevée entre 50 pixels et le 1/40ème de la plus petite dimension de l'image.

Le niveau de seuil de détection est relatif à la différence entre les valeurs maximales et minimales des données.

**Paramètres**

- **data** – Données d'entrée
- **size** – Taille du voisinage (par défaut : None)
- **level** – Niveau relatif (par défaut : 0.5)

**Renvoie**

Coordonnées des pics

`cdl.algorithms.image.get_contour_shapes(data : ndarray | MaskedArray, shape : Literal['circle', 'ellipse', 'polygon'] = 'ellipse', level : float = 0.5) → ndarray`

Trouve les contours de valeur iso dans un tableau 2D, au-dessus du niveau relatif (.5 signifie FWHM), puis ajuste les contours avec la forme ("ellipse" ou "cercle")

**Paramètres**

- **data** – Données d'entrée
- **shape** – Forme à ajuster. Par défaut : "ellipse"

— **level** – Niveau relatif (par défaut : 0.5)

#### Renvoie

Coordonnées des formes

```
cdl.algorithms.image.get_hough_circle_peaks(data : ndarray, min_radius : float | None = None,
                                             max_radius : float | None = None, nb_radius : int | None =
                                             None, min_distance : int = 1) → ndarray
```

Détecte les pics dans l'image à partir de la transformée de Hough du cercle, retourne les coordonnées du cercle.

#### Paramètres

- **data** – Données d'entrée
- **min\_radius** – Rayon minimum (par défaut : None)
- **max\_radius** – Rayon maximum (par défaut : None)
- **nb\_radius** – Nombre de rayons (par défaut : None)
- **min\_distance** – Distance minimale entre les cercles (par défaut : 1)

#### Renvoie

Coordonnées des cercles

```
cdl.algorithms.image.find_blobs_dog(data : ndarray, min_sigma : float = 1, max_sigma : float = 30,
                                     overlap : float = 0.5, threshold_rel : float = 0.2, exclude_border :
                                     bool = True) → ndarray
```

Recherche les taches dans l'image en niveaux de gris donnée en utilisant la méthode de différence de Gauss (DoG).

#### Paramètres

- **data** – L'image d'entrée en niveaux de gris.
- **min\_sigma** – Le rayon minimum de la tache en pixels.
- **max\_sigma** – Le rayon maximum de la tache en pixels.
- **overlap** – Le chevauchement minimum entre deux taches en pixels. Par exemple, si deux taches sont détectées avec des rayons de 10 et 12 respectivement, et que le chevauchement est défini sur 0,5, alors la zone de la plus petite tache sera ignorée et seule la zone de la plus grande tache sera retournée.
- **threshold\_rel** – La limite inférieure absolue pour les maxima de l'espace d'échelle. Les maxima locaux plus petits que **threshold\_rel** sont ignorés. Réduisez ceci pour détecter les taches avec moins d'intensité.
- **exclude\_border** – Si True, exclure les taches de la détection si elles sont trop proches du bord de l'image. La taille de la bordure est **min\_sigma**.

#### Renvoie

Coordonnées des taches

```
cdl.algorithms.image.find_blobs_doh(data : ndarray, min_sigma : float = 1, max_sigma : float = 30,
                                     overlap : float = 0.5, log_scale : bool = False, threshold_rel : float =
                                     0.2) → ndarray
```

Recherche les taches dans l'image en niveaux de gris donnée en utilisant la méthode du déterminant de Hessian (DoH).

#### Paramètres

- **data** – L'image d'entrée en niveaux de gris.
- **min\_sigma** – Le rayon minimum de la tache en pixels.
- **max\_sigma** – Le rayon maximum de la tache en pixels.
- **overlap** – Le chevauchement minimum entre deux taches en pixels. Par exemple, si deux taches sont détectées avec des rayons de 10 et 12 respectivement, et que le chevauchement est défini sur 0,5, alors la zone de la plus petite tache sera ignorée et seule la zone de la plus grande tache sera retournée.
- **log\_scale** – Si True, le rayon de chaque tache est retourné comme  $\sqrt{\text{sigma}}$  pour chaque tache détectée.

- **threshold\_rel** – La limite inférieure absolue pour les maxima de l'espace d'échelle. Les maxima locaux plus petits que **threshold\_rel** sont ignorés. Réduisez ceci pour détecter les taches avec moins d'intensité.

**Renvoie**

Coordonnées des taches

```
cdl.algorithms.image.find_blobs_log(data : ndarray, min_sigma : float = 1, max_sigma : float = 30,
                                   overlap : float = 0.5, log_scale : bool = False, threshold_rel : float =
                                   0.2, exclude_border : bool = True) → ndarray
```

Recherche les taches dans l'image en niveaux de gris donnée en utilisant la méthode du Laplacien de Gauss (LoG).

**Paramètres**

- **data** – L'image d'entrée en niveaux de gris.
- **min\_sigma** – Le rayon minimum de la tache en pixels.
- **max\_sigma** – Le rayon maximum de la tache en pixels.
- **overlap** – Le chevauchement minimum entre deux taches en pixels. Par exemple, si deux taches sont détectées avec des rayons de 10 et 12 respectivement, et que le chevauchement est défini sur 0,5, alors la zone de la plus petite tache sera ignorée et seule la zone de la plus grande tache sera retournée.
- **log\_scale** – Si True, le rayon de chaque tache est retourné comme  $\sqrt{\text{sigma}}$  pour chaque tache détectée.
- **threshold\_rel** – La limite inférieure absolue pour les maxima de l'espace d'échelle. Les maxima locaux plus petits que **threshold\_rel** sont ignorés. Réduisez ceci pour détecter les taches avec moins d'intensité.
- **exclude\_border** – Si True, exclure les taches de la détection si elles sont trop proches du bord de l'image. La taille de la bordure est **min\_sigma**.

**Renvoie**

Coordonnées des taches

```
cdl.algorithms.image.remove_overlapping_disks(coords : ndarray) → ndarray
```

Supprimer les disques qui se chevauchent parmi les coordonnées

**Paramètres**

- **coords** – Les coordonnées des disques

**Renvoie**

Les coordonnées des disques avec les disques qui se chevauchent supprimés

```
cdl.algorithms.image.find_blobs_opencv(data : ndarray, min_threshold : float | None = None,
                                       max_threshold : float | None = None, min_repeatability : int |
                                       None = None, min_dist_between_blobs : float | None = None,
                                       filter_by_color : bool | None = None, blob_color : int | None =
                                       None, filter_by_area : bool | None = None, min_area : float |
                                       None = None, max_area : float | None = None,
                                       filter_by_circularity : bool | None = None, min_circularity : float |
                                       None = None, max_circularity : float | None = None,
                                       filter_by_inertia : bool | None = None, min_inertia_ratio : float |
                                       None = None, max_inertia_ratio : float | None = None,
                                       filter_by_convexity : bool | None = None, min_convexity : float |
                                       None = None, max_convexity : float | None = None) → ndarray
```

Recherche les taches dans l'image en niveaux de gris donnée en utilisant le détecteur de taches simples d'OpenCV.

**Paramètres**

- **data** – L'image d'entrée en niveaux de gris.
- **min\_threshold** – L'intensité minimale de la tache.
- **max\_threshold** – L'intensité maximale de la tache.

- **min\_repeatability** – Le nombre minimum de fois qu’une tache est détectée avant qu’elle ne soit signalée.
- **min\_dist\_between\_blobs** – La distance minimale entre les taches.
- **filter\_by\_color** – Si True, les taches sont filtrées par couleur.
- **blob\_color** – La couleur des taches à filtrer par.
- **filter\_by\_area** – Si True, les taches sont filtrées par zone.
- **min\_area** – La zone minimale de la tache.
- **max\_area** – La zone maximale de la tache.
- **filter\_by\_circularity** – Si True, les taches sont filtrées par circularité.
- **min\_circularity** – La circularité minimale de la tache.
- **max\_circularity** – La circularité maximale de la tache.
- **filter\_by\_inertia** – Si True, les taches sont filtrées par inertie.
- **min\_inertia\_ratio** – Le rapport d’inertie minimale de la tache.
- **max\_inertia\_ratio** – Le rapport d’inertie maximale de la tache.
- **filter\_by\_convexity** – Si True, les taches sont filtrées par convexité.
- **min\_convexity** – La convexité minimale de la tache.
- **max\_convexity** – La convexité maximale de la tache.

**Renvoie**

Coordonnées des taches

### 3.1.3 Algorithmes de conversion de type de données

`cdl.algorithms.datatypes.is_integer_dtype(dtype : dtype) → bool`

Retourne True si le type de données est un type entier

**Paramètres****dtype** – Type de données à vérifier**Renvoie**

True si le type de données est un type entier

`cdl.algorithms.datatypes.is_complex_dtype(dtype : dtype) → bool`

Retourne True si le type de données est un type complexe

**Paramètres****dtype** – Type de données à vérifier**Renvoie**

True si le type de données est un type complexe

`cdl.algorithms.datatypes.clip_astype(data : ndarray, dtype : dtype) → ndarray`

Convert array to a new data type, after having clipped values to the new data type’s range if it is an integer type. If data type is not integer, this is equivalent to `data.astype(dtype)`.

**Paramètres**— **data** – Array to convert— **dtype** – Data type to convert to**Renvoie**

Array converted to new data type

### 3.1.4 Algorithmes de conversion de coordonnées

`cdl.algorithms.coordinates.circle_to_diameter(xc : float, yc : float, r : float) → tuple[float, float, float, float]`

Convertit le centre et le rayon du cercle en coordonnées de diamètre X

**Paramètres**

- **xc** – Abscisse du centre du cercle
- **yc** – Ordonnée du centre du cercle
- **r** – Rayon du cercle

**Renvoie**

Coordonnées de diamètre X du cercle

**Type renvoyé**

tuple

`cdl.algorithms.coordinates.array_circle_to_diameter(data : ndarray) → ndarray`

Convertit le centre et le rayon du cercle en coordonnées de diamètre X (version tableau)

**Paramètres**

**data** – Centre et rayon du cercle, sous la forme d'un tableau 2D (N, 3)

**Renvoie**

Coordonnées de diamètre X du cercle, sous la forme d'un tableau 2D (N, 4)

`cdl.algorithms.coordinates.circle_to_center_radius(x0 : float, y0 : float, x1 : float, y1 : float) → tuple[float, float, float]`

Convertit les coordonnées de diamètre X du cercle en centre et rayon

**Paramètres**

- **x0** – Abscisse de début de diamètre du cercle
- **y0** – Ordonnée de début de diamètre du cercle
- **x1** – Abscisse de fin de diamètre du cercle
- **y1** – Ordonnée de fin de diamètre du cercle

**Renvoie**

Centre et rayon du cercle

**Type renvoyé**

tuple

`cdl.algorithms.coordinates.array_circle_to_center_radius(data : ndarray) → ndarray`

Convertit les coordonnées de diamètre X du cercle en centre et rayon (version tableau)

**Paramètres**

**data** – Coordonnées de diamètre X du cercle, sous la forme d'un tableau 2D (N, 4)

**Renvoie**

Centre et rayon du cercle, sous la forme d'un tableau 2D (N, 3)

`cdl.algorithms.coordinates.ellipse_to_diameters(xc : float, yc : float, a : float, b : float, theta : float) → tuple[float, float, float, float, float, float, float, float]`

Convertit le centre, les axes et l'angle de l'ellipse en coordonnées de diamètres X/Y

**Paramètres**

- **xc** – Abscisse du centre de l'ellipse
- **yc** – Ordonnée du centre de l'ellipse
- **a** – Demi grand axe de l'ellipse
- **b** – Demi petit axe de l'ellipse
- **theta** – Angle de l'ellipse

**Renvoie**

Coordonnées des diamètres X/Y (grands/petits axes) de l'ellipse

`cdl.algorithms.coordinates.array_ellipse_to_diameters(data : ndarray) → ndarray`

Convertit le centre, les axes et l'angle de l'ellipse en coordonnées de diamètres X/Y (version tableau)

**Paramètres**

**data** – Centre, axes et angle de l'ellipse, sous la forme d'un tableau 2D (N, 5)

**Renvoie**

Coordonnées des diamètres X/Y de l'ellipse (grands/petits axes), sous la forme d'un tableau 2D (N, 8)

`cdl.algorithms.coordinates.ellipse_to_center_axes_angle(x0 : float, y0 : float, x1 : float, y1 : float, x2 : float, y2 : float, x3 : float, y3 : float) → tuple[float, float, float, float, float]`

Convertit les coordonnées de diamètres X/Y de l'ellipse en centre, axes et angle

**Paramètres**

- **x0** – abscisse de début du grand axe
- **y0** – ordonnée de début du grand axe
- **x1** – abscisse de fin du grand axe
- **y1** – ordonnée de fin du grand axe
- **x2** – abscisse de début du petit axe
- **y2** – ordonnée de début du petit axe
- **x3** – abscisse de fin du petit axe
- **y3** – ordonnée de fin du petit axe

**Renvoie**

Centre, axes et angle de l'ellipse

`cdl.algorithms.coordinates.array_ellipse_to_center_axes_angle(data : ndarray) → ndarray`

Convertit les coordonnées de diamètres X/Y de l'ellipse en centre, axes et angle (version tableau)

**Paramètres**

**data** – Coordonnées de diamètres X/Y de l'ellipse, sous la forme d'un tableau 2D (N, 8)

**Renvoie**

Centre, axes et angle de l'ellipse, sous la forme d'un tableau 2D (N, 5)

## 3.2 Paramètres (cdl.param)

Le module `cdl.param` fournit tous les paramètres de jeu de données utilisés par les modules `cdl.computation` et `cdl.core.gui.processor`.

Ces jeux de données sont définis dans d'autres modules :

- `cdl.computation.base`
- `cdl.computation.image`
- `cdl.computation.signal`

Le module `cdl.param` est donc un moyen pratique d'importer tous les ensembles de paramètres en une seule fois.

En fait, l'instruction d'importation suivante est équivalente à la précédente :

```
# Original import statement
from cdl.computation.base import MovingAverageParam
from cdl.computation.signal import PolynomialFitParam
from cdl.computation.image.exposure import EqualizeHistParam

# Equivalent import statement
from cdl.param import MovingAverageParam, PolynomialFitParam, EqualizeHistParam
```

### 3.2.1 Introduction aux jeux de données *DataSet*

Les jeux de données listés dans les sections suivantes sont utilisés pour définir les paramètres nécessaires aux diverses opérations de calcul et de traitement disponibles dans DataLab.

Ces jeux de données (sous-classes de `guidata.dataset.datatypes.Dataset`) sont définis dans d'autres modules :

Voici un exemple complet de la façon d'instancier un jeu de données et d'accéder à ses paramètres avec le jeu de données `cdl.param.BinningParam` :

```
class cdl.param.BinningParam
    Paramètres de binning

sx
    Nombre de pixels (X). Nombre de pixels adjacents à regrouper le long de l'axe des x. Entier
    supérieur à 2. Par défaut : 2.
        Type
            guidata.dataset.dataitems.IntItem

sy
    Nombre de pixels (Y). Nombre de pixels adjacents à regrouper le long de l'axe des y. Entier
    supérieur à 2. Par défaut : 2.
        Type
            guidata.dataset.dataitems.IntItem

operation
    Opération. Sélection unique parmi : "sum", "average", "median", "min", "max". Par défaut :
    "sum".
        Type
            guidata.dataset.dataitems.ChoiceItem

dtype_str
    Type de données. Type de données de l'image générée. Sélection unique parmi : "dtype",
    "float32", "float64", "complex128", "int32", "int16", "uint16", "uint8". Par défaut : "dtype".
        Type
            guidata.dataset.dataitems.ChoiceItem

change_pixel_size
    Modification de la taille des pixels de sorte que les dimensions de l'images restent les mêmes
    après l'opération. Par défaut : False.
        Type
            guidata.dataset.dataitems.BoolItem

classmethod create(sx : int, sy : int, operation : str, dtype_str : str, change_pixel_size : bool)
    → guidata.computation.image.BinningParam

    Renvoie une nouvelle instance de BinningParam avec les champs initialisés aux valeurs don-
    nées.

    Paramètres
    — sx (int) – Nombre de pixels (X). Nombre de pixels adjacents à regrouper le
      long de l'axe des x. Entier supérieur à 2. Par défaut : 2.
    — sy (int) – Nombre de pixels (Y). Nombre de pixels adjacents à regrouper le
      long de l'axe des y. Entier supérieur à 2. Par défaut : 2.
    — operation (str) – Opération. Sélection unique parmi : "sum", "average", "me-
      dian", "min", "max". Par défaut : "sum".
```

- **dtype\_str** (*str*) – Type de données. Type de données de l’image générée. Sélection unique parmi : “dtype”, “float32”, “float64”, “complex128”, “int32”, “int16”, “uint16”, “uint8”. Par défaut : “dtype”.
- **change\_pixel\_size** (*bool*) – Modification de la taille des pixels de sorte que les dimensions de l’images restent les mêmes après l’opération. Par défaut : False.

**Renvoie**

Nouvelle instance de BinningParam.

**Note :** Pour instancier un nouveau jeu de données BinningParam, vous pouvez utiliser la méthode de classe BinningParam.create() comme ceci :

```
BinningParam.create(sx=2, sy=2, operation='sum', dtype_str='dtype', change_
    pixel_size=False)
```

Vous pouvez également instancier un BinningParam par défaut puis initialiser les champs comme ceci :

```
param = BinningParam()
param.sx = 2
param.sy = 2
param.operation = 'sum'
param.dtype_str = 'dtype'
param.change_pixel_size = False
```

### 3.2.2 Paramètres communs

**class** cdl.param.ArithmeticParam

Arithmetic parameters

**operator**

Opérateur. Sélection unique parmi : “+”, “-”, “x”, “/”. Par défaut : “+”.

**Type**

guidata.dataset.dataitems.ChoiceItem

**factor**

Facteur. Par défaut : 1.0.

**Type**

guidata.dataset.dataitems.FloatItem

**constant**

Constante. Par défaut : 0.0.

**Type**

guidata.dataset.dataitems.FloatItem

**operation**

Opération. Par défaut : “”.

**Type**

guidata.dataset.dataitems.StringItem

**restore\_dtype**

Résultat. Par défaut : True.

**Type**

`guidata.dataset.dataitems.BoolItem`

**classmethod** **create**(*operator : str, factor : float, constant : float, operation : str, restore\_dtype : bool*) → *cdl.computation.base.ArithmeticParam*

Renvoie une nouvelle instance de `ArithmeticParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **operator** (*str*) – Opérateur. Sélection unique parmi : “+”, “-”, “×”, “/”. Par défaut : “+”.
- **factor** (*float*) – Facteur. Par défaut : 1.0.
- **constant** (*float*) – Constante. Par défaut : 0.0.
- **operation** (*str*) – Opération. Par défaut : “”.
- **restore\_dtype** (*bool*) – Résultat. Par défaut : True.

**Renvoie**

Nouvelle instance de `ArithmeticParam`.

**get\_operation**() → *str*

Return the operation string

**update\_operation**(*\_item, \_value*)

Update the operation item

**class** `cdl.param.ClipParam`

Paramètres d’écrtage

**lower**

Borne inférieure d’écrtage. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**upper**

Borne supérieure d’écrtage. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** **create**(*lower : float, upper : float*) → *cdl.computation.base.ClipParam*

Renvoie une nouvelle instance de `ClipParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **lower** (*float*) – Borne inférieure d’écrtage. Par défaut : None.
- **upper** (*float*) – Borne supérieure d’écrtage. Par défaut : None.

**Renvoie**

Nouvelle instance de `ClipParam`.

**class** `cdl.param.FFTParam`

Paramètres FFT

**shift**

Décaler la fréquence nulle au centre. Par défaut : None.

**Type**

`guidata.dataset.dataitems.BoolItem`

**classmethod** **create**(*shift : bool*) → *cdl.computation.base.FFTParam*

Renvoie une nouvelle instance de `FFTParam` avec les champs initialisés aux valeurs données.

**Paramètres****shift** (*bool*) – Décaler la fréquence nulle au centre. Par défaut : None.**Renvoie**

Nouvelle instance de FFTParam.

**class** `cdl.param.SpectrumParam`

Spectrum parameters

**log**

Par défaut : False.

**Type**`guidata.dataset.dataitems.BoolItem`**classmethod** `create(log : bool) → cdl.computation.base.SpectrumParam`Renvoie une nouvelle instance de `SpectrumParam` avec les champs initialisés aux valeurs données.**Paramètres****log** (*bool*) – Par défaut : False.**Renvoie**Nouvelle instance de `SpectrumParam`.**class** `cdl.param.GaussianParam`

Paramètres de filtre gaussien

**sigma**

. Par défaut : 1.0.

**Type**`guidata.dataset.dataitems.FloatItem`**classmethod** `create(sigma : float) → cdl.computation.base.GaussianParam`Renvoie une nouvelle instance de `GaussianParam` avec les champs initialisés aux valeurs données.**Paramètres****sigma** (*float*) – . Par défaut : 1.0.**Renvoie**Nouvelle instance de `GaussianParam`.**class** `cdl.param.MovingAverageParam`

Paramètres de moyenne mobile

**n**

Taille de la fenêtre glissante. Entier supérieur à 1. Par défaut : 3.

**Type**`guidata.dataset.dataitems.IntItem`**mode**

Mode du filtre : - “reflect” : réfléchir les données à la frontière - “constant” : remplir avec une valeur constante - “nearest” : remplir avec la valeur la plus proche - “mirror” : réfléchir les données à la frontière avec les données elles-mêmes - “wrap” : frontière circulaire. Sélection unique parmi : “reflect”, “constant”, “nearest”, “mirror”, “wrap”. Par défaut : “reflect”.

**Type**`guidata.dataset.dataitems.ChoiceItem`**classmethod** `create(n : int, mode : str) → cdl.computation.base.MovingAverageParam`Renvoie une nouvelle instance de `MovingAverageParam` avec les champs initialisés aux valeurs données.**Paramètres**

- **n** (*int*) – Taille de la fenêtre glissante. Entier supérieur à 1. Par défaut : 3.
- **mode** (*str*) – Mode du filtre : - “reflect” : réfléchir les données à la frontière - “constant” : remplir avec une valeur constante - “nearest” : remplir avec la valeur la plus proche - “mirror” : réfléchir les données à la frontière avec les données elles-mêmes - “wrap” : frontière circulaire. Sélection unique parmi : “reflect”, “constant”, “nearest”, “mirror”, “wrap”. Par défaut : “reflect”.

**Renvoie**

Nouvelle instance de `MovingAverageParam`.

**class** `cdl.param.MovingMedianParam`

Paramètres de médiane mobile

**n**

Taille de la fenêtre glissante. Entier supérieur à 1, impair. Par défaut : 3.

**Type**

`guidata.dataset.dataitems.IntItem`

**mode**

Mode du filtre : - “reflect” : réfléchir les données à la frontière - “constant” : remplir avec une valeur constante - “nearest” : remplir avec la valeur la plus proche - “mirror” : réfléchir les données à la frontière avec les données elles-mêmes - “wrap” : frontière circulaire. Sélection unique parmi : “reflect”, “constant”, “nearest”, “mirror”, “wrap”. Par défaut : “nearest”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**classmethod** `create(n : int, mode : str) → cdl.computation.base.MovingMedianParam`

Renvoie une nouvelle instance de `MovingMedianParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **n** (*int*) – Taille de la fenêtre glissante. Entier supérieur à 1, impair. Par défaut : 3.
- **mode** (*str*) – Mode du filtre : - “reflect” : réfléchir les données à la frontière - “constant” : remplir avec une valeur constante - “nearest” : remplir avec la valeur la plus proche - “mirror” : réfléchir les données à la frontière avec les données elles-mêmes - “wrap” : frontière circulaire. Sélection unique parmi : “reflect”, “constant”, “nearest”, “mirror”, “wrap”. Par défaut : “nearest”.

**Renvoie**

Nouvelle instance de `MovingMedianParam`.

**class** `cdl.param.ConstantParam`

Parameter used to set a constant value to used in operations

**value**

Constante. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(value : float) → cdl.computation.base.ConstantParam`

Renvoie une nouvelle instance de `ConstantParam` avec les champs initialisés aux valeurs données.

**Paramètres**

**value** (*float*) – Constante. Par défaut : None.

**Renvoie**

Nouvelle instance de `ConstantParam`.

### 3.2.3 Paramètres des signaux

#### **class** `cdl.param.DataTypeSParam`

Convertir les paramètres de type de données du signal

##### **dtype\_str**

Type de données de destination. Type de données de l'image générée. Sélection unique parmi : "float32", "float64", "complex128". Par défaut : "float32".

##### **Type**

`guidata.dataset.dataitems.ChoiceItem`

#### **classmethod** `create(dtype_str : str) → cdl.computation.signal.DataTypeSParam`

Renvoie une nouvelle instance de `DataTypeSParam` avec les champs initialisés aux valeurs données.

##### **Paramètres**

**dtype\_str** (*str*) – Type de données de destination. Type de données de l'image générée. Sélection unique parmi : "float32", "float64", "complex128". Par défaut : "float32".

##### **Renvoie**

Nouvelle instance de `DataTypeSParam`.

#### **class** `cdl.param.FWHMParam`

Paramètres de LMH

##### **method**

Méthode. Sélection unique parmi : "zero-crossing", "gauss", "lorentz", "voigt". Par défaut : "zero-crossing".

##### **Type**

`guidata.dataset.dataitems.ChoiceItem`

##### **xmin**

$X_{\text{MIN}}$ . Borne x inférieure (vide pour aucune limite, c'est-à-dire le début du signal). Par défaut : None.

##### **Type**

`guidata.dataset.dataitems.FloatItem`

##### **xmax**

$X_{\text{MAX}}$ . Borne x supérieure (vide pour aucune limite, c'est-à-dire la fin du signal). Par défaut : None.

##### **Type**

`guidata.dataset.dataitems.FloatItem`

#### **classmethod** `create(method : str, xmin : float, xmax : float) → cdl.computation.signal.FWHMParam`

Renvoie une nouvelle instance de `FWHMParam` avec les champs initialisés aux valeurs données.

##### **Paramètres**

- **method** (*str*) – Méthode. Sélection unique parmi : "zero-crossing", "gauss", "lorentz", "voigt". Par défaut : "zero-crossing".
- **xmin** (*float*) –  $X_{\text{MIN}}$ . Borne x inférieure (vide pour aucune limite, c'est-à-dire le début du signal). Par défaut : None.
- **xmax** (*float*) –  $X_{\text{MAX}}$ . Borne x supérieure (vide pour aucune limite, c'est-à-dire la fin du signal). Par défaut : None.

##### **Renvoie**

Nouvelle instance de `FWHMParam`.

#### **class** `cdl.param.NormalizeParam`

Paramètres de normalisation

**method**

Normaliser par rapport à. Sélection unique parmi : “maximum”, “amplitude”, “area”, “energy”, “rms”. Par défaut : “maximum”.

**Type**

[guidata.dataset.dataitems.ChoiceItem](#)

**classmethod** **create**(*method* : *str*) → *cdl.computation.base.NormalizeParam*

Renvoie une nouvelle instance de `NormalizeParam` avec les champs initialisés aux valeurs données.

**Paramètres**

**method** (*str*) – Normaliser par rapport à. Sélection unique parmi : “maximum”, “amplitude”, “area”, “energy”, “rms”. Par défaut : “maximum”.

**Renvoie**

Nouvelle instance de `NormalizeParam`.

**class** `cdl.param.PeakDetectionParam`

Paramètres de détection de pics

**threshold**

Seuil. Entier compris entre 0 et 100, unité : %. Par défaut : 30.

**Type**

[guidata.dataset.dataitems.IntItem](#)

**min\_dist**

Distance minimale. Entier supérieur à 1, unité : points. Par défaut : 1.

**Type**

[guidata.dataset.dataitems.IntItem](#)

**classmethod** **create**(*threshold* : *int*, *min\_dist* : *int*) → *cdl.computation.signal.PeakDetectionParam*

Renvoie une nouvelle instance de `PeakDetectionParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **threshold** (*int*) – Seuil. Entier compris entre 0 et 100, unité : %. Par défaut : 30.
- **min\_dist** (*int*) – Distance minimale. Entier supérieur à 1, unité : points. Par défaut : 1.

**Renvoie**

Nouvelle instance de `PeakDetectionParam`.

**class** `cdl.param.PolynomialFitParam`

Paramètres de régression polynomiale

**degree**

Degré. Entier compris entre 1 et 10. Par défaut : 3.

**Type**

[guidata.dataset.dataitems.IntItem](#)

**classmethod** **create**(*degree* : *int*) → *cdl.computation.signal.PolynomialFitParam*

Renvoie une nouvelle instance de `PolynomialFitParam` avec les champs initialisés aux valeurs données.

**Paramètres**

**degree** (*int*) – Degré. Entier compris entre 1 et 10. Par défaut : 3.

**Renvoie**

Nouvelle instance de `PolynomialFitParam`.

**class** `cdl.param.XYCalibrateParam`

Paramètres de calibration du signal

**axis**

Étalonner. Sélection unique parmi : “x”, “y”. Par défaut : “y”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**a**

Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**b**

Par défaut : 0.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(axis : str, a : float, b : float) → cdl.computation.signal.XYCalibrateParam`

Renvoie une nouvelle instance de `XYCalibrateParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **axis** (*str*) – Étalonner. Sélection unique parmi : “x”, “y”. Par défaut : “y”.
- **a** (*float*) – Par défaut : 1.0.
- **b** (*float*) – Par défaut : 0.0.

**Renvoie**

Nouvelle instance de `XYCalibrateParam`.

**class** `cdl.param.InterpolationParam`

Paramètres d’interpolation

**method**

Méthode d’interpolation. Sélection unique parmi : “linear”, “spline”, “quadratic”, “cubic”, “barycentric”, “pchip”. Par défaut : “linear”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**fill\_value**

Valeur de remplissage. Valeur à utiliser pour les points en dehors du domaine d’interpolation (utilisé uniquement avec les méthodes linéaire, cubique et pchip). Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(method : str, fill_value : float) → cdl.computation.signal.InterpolationParam`

Renvoie une nouvelle instance de `InterpolationParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **method** (*str*) – Méthode d’interpolation. Sélection unique parmi : “linear”, “spline”, “quadratic”, “cubic”, “barycentric”, “pchip”. Par défaut : “linear”.
- **fill\_value** (*float*) – Valeur de remplissage. Valeur à utiliser pour les points en dehors du domaine d’interpolation (utilisé uniquement avec les méthodes linéaire, cubique et pchip). Par défaut : None.

**Renvoie**

Nouvelle instance de `InterpolationParam`.

**class** `cdl.param.ResamplingParam`

Paramètres de rééchantillonnage

**method**

Méthode d'interpolation. Sélection unique parmi : “linear”, “spline”, “quadratic”, “cubic”, “barycentric”, “pchip”. Par défaut : “linear”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**fill\_value**

Valeur de remplissage. Valeur à utiliser pour les points en dehors du domaine d'interpolation (utilisé uniquement avec les méthodes linéaire, cubique et pchip). Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**xmin**

$X_{\min}$ . Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**xmax**

$X_{\max}$ . Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**mode**

Sélection unique parmi : “dx”, “nbpts”. Par défaut : “nbpts”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**dx**

X. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**nbpts**

Nombre de points. Par défaut : None.

**Type**

`guidata.dataset.dataitems.IntItem`

**classmethod create**(*method* : *str*, *fill\_value* : *float*, *xmin* : *float*, *xmax* : *float*, *mode* : *str*, *dx* : *float*, *nbpts* : *int*) → *cdl.computation.signal.ResamplingParam*

Renvoie une nouvelle instance de `ResamplingParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **method** (*str*) – Méthode d'interpolation. Sélection unique parmi : “linear”, “spline”, “quadratic”, “cubic”, “barycentric”, “pchip”. Par défaut : “linear”.
- **fill\_value** (*float*) – Valeur de remplissage. Valeur à utiliser pour les points en dehors du domaine d'interpolation (utilisé uniquement avec les méthodes linéaire, cubique et pchip). Par défaut : None.
- **xmin** (*float*) –  $X_{\min}$ . Par défaut : None.
- **xmax** (*float*) –  $X_{\max}$ . Par défaut : None.
- **mode** (*str*) – Sélection unique parmi : “dx”, “nbpts”. Par défaut : “nbpts”.
- **dx** (*float*) – X. Par défaut : None.
- **nbpts** (*int*) – Nombre de points. Par défaut : None.

**Renvoie**

Nouvelle instance de `ResamplingParam`.

**class** `cdl.param.DetrendingParam`

Elimination de tendance

**method**

Méthode d'élimination de la tendance. Sélection unique parmi : "linear", "constant". Par défaut : "linear".

**Type**`guidata.dataset.dataitems.ChoiceItem`**classmethod** `create(method : str) → cdl.computation.signal.DetrendingParam`Renvoie une nouvelle instance de `DetrendingParam` avec les champs initialisés aux valeurs données.**Paramètres****method** (*str*) – Méthode d'élimination de la tendance. Sélection unique parmi : "linear", "constant". Par défaut : "linear".**Renvoie**Nouvelle instance de `DetrendingParam`.**class** `cdl.param.PowerParam`

Power parameters

**power**

Puissance. Par défaut : 2.0.

**Type**`guidata.dataset.dataitems.FloatItem`**classmethod** `create(power : float) → cdl.computation.signal.PowerParam`Renvoie une nouvelle instance de `PowerParam` avec les champs initialisés aux valeurs données.**Paramètres****power** (*float*) – Puissance. Par défaut : 2.0.**Renvoie**Nouvelle instance de `PowerParam`.**class** `cdl.param.WindowingParam`

Windowing parameters

**method**

Méthode. Sélection unique parmi : "barthann", "bartlett", "blackman", "blackman-harris", "bohman", "boxcar", "cosine", "exponential", "flat-top", "gaussian", "hamming", "hanning", "kaiser", "lanczos", "nuttall", "parzen", "rectangular", "taylor", "tukey". Par défaut : "hamming".

**Type**`guidata.dataset.dataitems.ChoiceItem`**alpha**

Paramètre de forme de la fonction de fenêtrage de tukey. Par défaut : 0.5.

**Type**`guidata.dataset.dataitems.FloatItem`**beta**

Paramètre de forme de la fonction de fenêtrage de kaiser. Par défaut : 14.0.

**Type**`guidata.dataset.dataitems.FloatItem`**sigma**

Paramètre de forme de la fonction de fenêtrage gaussienne. Par défaut : 0.5.

**Type**`guidata.dataset.dataitems.FloatItem`

**classmethod create**(*method* : *str*, *alpha* : *float*, *beta* : *float*, *sigma* : *float*) →  
*cdl.computation.signal.WindowingParam*

Renvoie une nouvelle instance de `WindowingParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **method** (*str*) – Méthode. Sélection unique parmi : “barthann”, “bartlett”, “blackman”, “blackman-harris”, “bohman”, “boxcar”, “cosine”, “exponential”, “flat-top”, “gaussian”, “hamming”, “hanning”, “kaiser”, “lanczos”, “nuttall”, “parzen”, “rectangular”, “taylor”, “tukey”. Par défaut : “hamming”.
- **alpha** (*float*) – Paramètre de forme de la fonction de fenêtrage de tukey. Par défaut : 0.5.
- **beta** (*float*) – Paramètre de forme de la fonction de fenêtrage de kaiser. Par défaut : 14.0.
- **sigma** (*float*) – Paramètre de forme de la fonction de fenêtrage gaussienne. Par défaut : 0.5.

**Renvoie**

Nouvelle instance de `WindowingParam`.

**class** `cdl.param.LowPassFilterParam`

Low-pass filter parameters

**method**

Méthode de filtrage. Sélection unique parmi : “bessel”, “butter”, “cheby1”, “cheby2”, “ellip”. Par défaut : “bessel”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**order**

Ordre du filtre. Entier supérieur à 1. Par défaut : 3.

**Type**

`guidata.dataset.dataitems.IntItem`

**f\_cut0**

Fréquence de coupure basse. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**f\_cut1**

Fréquence de coupure haute. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**rp**

Ondulation de la bande passante. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Type**

`guidata.dataset.dataitems.FloatItem`

**rs**

Atténuation de la bande d’arrêt. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod create**(*method* : *str*, *order* : *int*, *f\_cut0* : *float*, *f\_cut1* : *float*, *rp* : *float*, *rs* : *float*) →  
*cdl.computation.signal.LowPassFilterParam*

Renvoie une nouvelle instance de `LowPassFilterParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **method** (*str*) – Méthode de filtrage. Sélection unique parmi : “bessel”, “butter”, “cheby1”, “cheby2”, “ellip”. Par défaut : “bessel”.
- **order** (*int*) – Ordre du filtre. Entier supérieur à 1. Par défaut : 3.
- **f\_cut0** (*float*) – Fréquence de coupure basse. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.
- **f\_cut1** (*float*) – Fréquence de coupure haute. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.
- **rp** (*float*) – Ondulation de la bande passante. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.
- **rs** (*float*) – Atténuation de la bande d’arrêt. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Renvoie**

Nouvelle instance de `LowPassFilterParam`.

**class** `cdl.param.HighPassFilterParam`

High-pass filter parameters

**method**

Méthode de filtrage. Sélection unique parmi : “bessel”, “butter”, “cheby1”, “cheby2”, “ellip”. Par défaut : “bessel”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**order**

Ordre du filtre. Entier supérieur à 1. Par défaut : 3.

**Type**

`guidata.dataset.dataitems.IntItem`

**f\_cut0**

Fréquence de coupure basse. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**f\_cut1**

Fréquence de coupure haute. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**rp**

Ondulation de la bande passante. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Type**

`guidata.dataset.dataitems.FloatItem`

**rs**

Atténuation de la bande d’arrêt. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create`(*method* : *str*, *order* : *int*, *f\_cut0* : *float*, *f\_cut1* : *float*, *rp* : *float*, *rs* : *float*) → `cdl.computation.signal.HighPassFilterParam`

Renvoie une nouvelle instance de `HighPassFilterParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **method** (*str*) – Méthode de filtrage. Sélection unique parmi : “bessel”, “butter”, “cheby1”, “cheby2”, “ellip”. Par défaut : “bessel”.

- **order** (*int*) – Ordre du filtre. Entier supérieur à 1. Par défaut : 3.
- **f\_cut0** (*float*) – Fréquence de coupure basse. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.
- **f\_cut1** (*float*) – Fréquence de coupure haute. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.
- **rp** (*float*) – Ondulation de la bande passante. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.
- **rs** (*float*) – Atténuation de la bande d’arrêt. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Renvoie**

Nouvelle instance de `HighPassFilterParam`.

**class** `cdl.param.BandPassFilterParam`

Band-pass filter parameters

**method**

Méthode de filtrage. Sélection unique parmi : “bessel”, “butter”, “cheby1”, “cheby2”, “ellip”. Par défaut : “bessel”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**order**

Ordre du filtre. Entier supérieur à 1. Par défaut : 3.

**Type**

`guidata.dataset.dataitems.IntItem`

**f\_cut0**

Fréquence de coupure basse. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**f\_cut1**

Fréquence de coupure haute. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**rp**

Ondulation de la bande passante. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Type**

`guidata.dataset.dataitems.FloatItem`

**rs**

Atténuation de la bande d’arrêt. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** **create**(*method* : *str*, *order* : *int*, *f\_cut0* : *float*, *f\_cut1* : *float*, *rp* : *float*, *rs* : *float*) → `cdl.computation.signal.BandPassFilterParam`

Renvoie une nouvelle instance de `BandPassFilterParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **method** (*str*) – Méthode de filtrage. Sélection unique parmi : “bessel”, “butter”, “cheby1”, “cheby2”, “ellip”. Par défaut : “bessel”.
- **order** (*int*) – Ordre du filtre. Entier supérieur à 1. Par défaut : 3.
- **f\_cut0** (*float*) – Fréquence de coupure basse. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.

- **f\_cut1** (*float*) – Fréquence de coupure haute. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.
- **rp** (*float*) – Ondulation de la bande passante. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.
- **rs** (*float*) – Atténuation de la bande d’arrêt. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Renvoie**

Nouvelle instance de `BandPassFilterParam`.

**class** `cdl.param.BandStopFilterParam`

Band-stop filter parameters

**method**

Méthode de filtrage. Sélection unique parmi : “bessel”, “butter”, “cheby1”, “cheby2”, “ellip”. Par défaut : “bessel”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**order**

Ordre du filtre. Entier supérieur à 1. Par défaut : 3.

**Type**

`guidata.dataset.dataitems.IntItem`

**f\_cut0**

Fréquence de coupure basse. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**f\_cut1**

Fréquence de coupure haute. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**rp**

Ondulation de la bande passante. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Type**

`guidata.dataset.dataitems.FloatItem`

**rs**

Atténuation de la bande d’arrêt. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create`(*method* : *str*, *order* : *int*, *f\_cut0* : *float*, *f\_cut1* : *float*, *rp* : *float*, *rs* : *float*) → `cdl.computation.signal.BandStopFilterParam`

Renvoie une nouvelle instance de `BandStopFilterParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **method** (*str*) – Méthode de filtrage. Sélection unique parmi : “bessel”, “butter”, “cheby1”, “cheby2”, “ellip”. Par défaut : “bessel”.
- **order** (*int*) – Ordre du filtre. Entier supérieur à 1. Par défaut : 3.
- **f\_cut0** (*float*) – Fréquence de coupure basse. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.
- **f\_cut1** (*float*) – Fréquence de coupure haute. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.

- **rp** (*float*) – Ondulation de la bande passante. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.
- **rs** (*float*) – Atténuation de la bande d’arrêt. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Renvoie**

Nouvelle instance de `BandStopFilterParam`.

**class** `cdl.param.DynamicParam`

Parameters for dynamic range computation (ENOB, SNR, SINAD, THD, SFDR)

**full\_scale**

Pleine échelle. Flottant supérieur à 0.0, unité : v. Par défaut : 0.16.

**Type**

`guidata.dataset.dataitems.FloatItem`

**unit**

Unité. Unité pour sinad. Sélection unique parmi : “dBc”, “dBFS”. Par défaut : “dBc”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**nb\_harm**

Nombre d’harmoniques. Nombre d’harmoniques à considérer pour le thd. Entier supérieur à 1. Par défaut : 5.

**Type**

`guidata.dataset.dataitems.IntItem`

**classmethod** `create`(*full\_scale* : *float*, *unit* : *str*, *nb\_harm* : *int*) → `cdl.computation.signal.DynamicParam`

Renvoie une nouvelle instance de `DynamicParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **full\_scale** (*float*) – Pleine échelle. Flottant supérieur à 0.0, unité : v. Par défaut : 0.16.
- **unit** (*str*) – Unité. Unité pour sinad. Sélection unique parmi : “dBc”, “dBFS”. Par défaut : “dBc”.
- **nb\_harm** (*int*) – Nombre d’harmoniques. Nombre d’harmoniques à considérer pour le thd. Entier supérieur à 1. Par défaut : 5.

**Renvoie**

Nouvelle instance de `DynamicParam`.

### 3.2.4 Paramètres des images

#### Paramètres de base des images

**class** `cdl.param.BinningParam`

Paramètres de binning

**sx**

Nombre de pixels (X). Nombre de pixels adjacents à regrouper le long de l’axe des x. Entier supérieur à 2. Par défaut : 2.

**Type**

`guidata.dataset.dataitems.IntItem`

**sy**

Nombre de pixels (Y). Nombre de pixels adjacents à regrouper le long de l’axe des y. Entier supérieur à 2. Par défaut : 2.

**Type**`guidata.dataset.dataitems.IntItem`**operation**

Opération. Sélection unique parmi : “sum”, “average”, “median”, “min”, “max”. Par défaut : “sum”.

**Type**`guidata.dataset.dataitems.ChoiceItem`**dtype\_str**

Type de données. Type de données de l’image générée. Sélection unique parmi : “dtype”, “float32”, “float64”, “complex128”, “int32”, “int16”, “uint16”, “uint8”. Par défaut : “dtype”.

**Type**`guidata.dataset.dataitems.ChoiceItem`**change\_pixel\_size**

Modification de la taille des pixels de sorte que les dimensions de l’images restent les mêmes après l’opération. Par défaut : False.

**Type**`guidata.dataset.dataitems.BoolItem`

**classmethod create**(*sx : int, sy : int, operation : str, dtype\_str : str, change\_pixel\_size : bool*) → *cdl.computation.image.BinningParam*

Renvoie une nouvelle instance de `BinningParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **sx** (*int*) – Nombre de pixels (X). Nombre de pixels adjacents à regrouper le long de l’axe des x. Entier supérieur à 2. Par défaut : 2.
- **sy** (*int*) – Nombre de pixels (Y). Nombre de pixels adjacents à regrouper le long de l’axe des y. Entier supérieur à 2. Par défaut : 2.
- **operation** (*str*) – Opération. Sélection unique parmi : “sum”, “average”, “median”, “min”, “max”. Par défaut : “sum”.
- **dtype\_str** (*str*) – Type de données. Type de données de l’image générée. Sélection unique parmi : “dtype”, “float32”, “float64”, “complex128”, “int32”, “int16”, “uint16”, “uint8”. Par défaut : “dtype”.
- **change\_pixel\_size** (*bool*) – Modification de la taille des pixels de sorte que les dimensions de l’images restent les mêmes après l’opération. Par défaut : False.

**Renvoie**

Nouvelle instance de `BinningParam`.

**class** `cdl.param.ButterworthParam`

Paramètres de filtre Butterworth

**cut\_off**

Fréquence de coupure relative. Fréquence de coupure relative. Flottant compris entre 0.0 et 0.5. Par défaut : 0.005.

**Type**`guidata.dataset.dataitems.FloatItem`**high\_pass**

Si vrai, appliquer un filtre passe-haut au lieu d’un filtre passe-bas. Par défaut : False.

**Type**`guidata.dataset.dataitems.BoolItem`**order**

Ordre. Ordre du filtre de butterworth. Entier supérieur à 1. Par défaut : 2.

**Type**`guidata.dataset.dataitems.IntItem`

**classmethod create**(*cut\_off* : *float*, *high\_pass* : *bool*, *order* : *int*) →  
*cdl.computation.image.ButterworthParam*

Renvoie une nouvelle instance de `ButterworthParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **cut\_off** (*float*) – Fréquence de coupure relative. Fréquence de coupure relative. Flottant compris entre 0.0 et 0.5. Par défaut : 0.005.
- **high\_pass** (*bool*) – Si vrai, appliquer un filtre passe-haut au lieu d’un filtre passe-bas. Par défaut : False.
- **order** (*int*) – Ordre. Ordre du filtre de butterworth. Entier supérieur à 1. Par défaut : 2.

**Renvoie**

Nouvelle instance de `ButterworthParam`.

**class** `cdl.param.DataTypeIParam`

Convertir les paramètres de type de données de l’image

**dtype\_str**

Type de données de destination. Type de données de l’image générée. Sélection unique parmi : “float32”, “float64”, “complex128”, “int32”, “int16”, “uint16”, “uint8”. Par défaut : “float32”.

**Type**`guidata.dataset.dataitems.ChoiceItem`

**classmethod create**(*dtype\_str* : *str*) → *cdl.computation.image.DataTypeIParam*

Renvoie une nouvelle instance de `DataTypeIParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- dtype\_str** (*str*) – Type de données de destination. Type de données de l’image générée. Sélection unique parmi : “float32”, “float64”, “complex128”, “int32”, “int16”, “uint16”, “uint8”. Par défaut : “float32”.

**Renvoie**

Nouvelle instance de `DataTypeIParam`.

**class** `cdl.param.FlatFieldParam`

Paramètres de correction de champ plat

**threshold**

Seuil. Par défaut : 0.0.

**Type**`guidata.dataset.dataitems.FloatItem`

**classmethod create**(*threshold* : *float*) → *cdl.computation.image.FlatFieldParam*

Renvoie une nouvelle instance de `FlatFieldParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- threshold** (*float*) – Seuil. Par défaut : 0.0.

**Renvoie**

Nouvelle instance de `FlatFieldParam`.

**class** `cdl.param.GridParam`

Paramètres de grille

**direction**

Distribuer selon les. Sélection unique parmi : “col”, “row”. Par défaut : “col”.

**Type**`guidata.dataset.dataitems.ChoiceItem`

**cols**

Colonnes. Entier, non nul. Par défaut : 1.

**Type**

`guidata.dataset.dataitems.IntItem`

**rows**

Lignes. Entier, non nul. Par défaut : 1.

**Type**

`guidata.dataset.dataitems.IntItem`

**colspac**

Espace entre chaque colonne. Flottant supérieur à 0.0. Par défaut : 0.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**rowspac**

Espace entre chaque ligne. Flottant supérieur à 0.0. Par défaut : 0.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod create**(*direction* : *str*, *cols* : *int*, *rows* : *int*, *colspac* : *float*, *rowspac* : *float*) →  
*cdl.computation.image.GridParam*

Renvoie une nouvelle instance de `GridParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **direction** (*str*) – Distribuer selon les. Sélection unique parmi : “col”, “row”. Par défaut : “col”.
- **cols** (*int*) – Colonnes. Entier, non nul. Par défaut : 1.
- **rows** (*int*) – Lignes. Entier, non nul. Par défaut : 1.
- **colspac** (*float*) – Espace entre chaque colonne. Flottant supérieur à 0.0. Par défaut : 0.0.
- **rowspac** (*float*) – Espace entre chaque ligne. Flottant supérieur à 0.0. Par défaut : 0.0.

**Renvoie**

Nouvelle instance de `GridParam`.

**class cdl.param.HoughCircleParam**

Paramètres de transformation de Hough circulaire

**min\_radius**

Rayon<sub>min</sub>. Entier supérieur à 0, non nul, unité : pixels. Par défaut : None.

**Type**

`guidata.dataset.dataitems.IntItem`

**max\_radius**

Rayon<sub>max</sub>. Entier supérieur à 0, non nul, unité : pixels. Par défaut : None.

**Type**

`guidata.dataset.dataitems.IntItem`

**min\_distance**

Distance minimale. Entier supérieur à 0. Par défaut : None.

**Type**

`guidata.dataset.dataitems.IntItem`

**classmethod create**(*min\_radius* : *int*, *max\_radius* : *int*, *min\_distance* : *int*) →  
*cdl.computation.image.HoughCircleParam*

Renvoie une nouvelle instance de `HoughCircleParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **min\_radius** (*int*) – Rayon<sub>min</sub>. Entier supérieur à 0, non nul, unité : pixels. Par défaut : None.
- **max\_radius** (*int*) – Rayon<sub>max</sub>. Entier supérieur à 0, non nul, unité : pixels. Par défaut : None.
- **min\_distance** (*int*) – Distance minimale. Entier supérieur à 0. Par défaut : None.

**Renvoie**

Nouvelle instance de `HoughCircleParam`.

**class** `cdl.param.LogP1Param`

Paramètres de log10

**n**

Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(n : float) → cdl.computation.image.LogP1Param`

Renvoie une nouvelle instance de `LogP1Param` avec les champs initialisés aux valeurs données.

**Paramètres**

- n** (*float*) – Par défaut : None.

**Renvoie**

Nouvelle instance de `LogP1Param`.

**class** `cdl.param.LineProfileParam`

Paramètres de profil horizontal ou vertical

**direction**

Sélection unique parmi : “horizontal”, “vertical”. Par défaut : “horizontal”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**row**

Ligne. Entier supérieur à 0. Par défaut : 0.

**Type**

`guidata.dataset.dataitems.IntItem`

**col**

Colonne. Entier supérieur à 0. Par défaut : 0.

**Type**

`guidata.dataset.dataitems.IntItem`

**classmethod** `create(direction : str, row : int, col : int) → cdl.computation.image.LineProfileParam`

Renvoie une nouvelle instance de `LineProfileParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **direction** (*str*) – Sélection unique parmi : “horizontal”, “vertical”. Par défaut : “horizontal”.
- **row** (*int*) – Ligne. Entier supérieur à 0. Par défaut : 0.
- **col** (*int*) – Colonne. Entier supérieur à 0. Par défaut : 0.

**Renvoie**

Nouvelle instance de `LineProfileParam`.

**class** `cdl.param.SegmentProfileParam`

Segment profile parameters

**row1**

Ligne (début). Entier supérieur à 0. Par défaut : 0.

**Type**`guidata.dataset.dataitems.IntItem`**col1**

Colonne (début). Entier supérieur à 0. Par défaut : 0.

**Type**`guidata.dataset.dataitems.IntItem`**row2**

Ligne (fin). Entier supérieur à 0. Par défaut : 0.

**Type**`guidata.dataset.dataitems.IntItem`**col2**

Colonne (fin). Entier supérieur à 0. Par défaut : 0.

**Type**`guidata.dataset.dataitems.IntItem`**classmethod** `create(row1 : int, col1 : int, row2 : int, col2 : int) →``cdl.computation.image.SegmentProfileParam`Renvoie une nouvelle instance de `SegmentProfileParam` avec les champs initialisés aux valeurs données.**Paramètres**

- **row1** (*int*) – Ligne (début). Entier supérieur à 0. Par défaut : 0.
- **col1** (*int*) – Colonne (début). Entier supérieur à 0. Par défaut : 0.
- **row2** (*int*) – Ligne (fin). Entier supérieur à 0. Par défaut : 0.
- **col2** (*int*) – Colonne (fin). Entier supérieur à 0. Par défaut : 0.

**Renvoie**Nouvelle instance de `SegmentProfileParam`.**class** `cdl.param.AverageProfileParam`

Paramètres de profil horizontal ou vertical moyen

**direction**

Sélection unique parmi : “horizontal”, “vertical”. Par défaut : “horizontal”.

**Type**`guidata.dataset.dataitems.ChoiceItem`**row1**

Ligne 1. Entier supérieur à 0. Par défaut : 0.

**Type**`guidata.dataset.dataitems.IntItem`**row2**

Ligne 2. Entier supérieur à -1. Par défaut : -1.

**Type**`guidata.dataset.dataitems.IntItem`**col1**

Colonne 1. Entier supérieur à 0. Par défaut : 0.

**Type**`guidata.dataset.dataitems.IntItem`**col2**

Colonne 2. Entier supérieur à -1. Par défaut : -1.

**Type**`guidata.dataset.dataitems.IntItem`

**classmethod create**(*direction* : *str*, *row1* : *int*, *row2* : *int*, *col1* : *int*, *col2* : *int*) → `cdl.computation.image.AverageProfileParam`

Renvoie une nouvelle instance de `AverageProfileParam` avec les champs initialisés aux valeurs données.**Paramètres**

- **direction** (*str*) – Sélection unique parmi : “horizontal”, “vertical”. Par défaut : “horizontal”.
- **row1** (*int*) – Ligne 1. Entier supérieur à 0. Par défaut : 0.
- **row2** (*int*) – Ligne 2. Entier supérieur à -1. Par défaut : -1.
- **col1** (*int*) – Colonne 1. Entier supérieur à 0. Par défaut : 0.
- **col2** (*int*) – Colonne 2. Entier supérieur à -1. Par défaut : -1.

**Renvoie**Nouvelle instance de `AverageProfileParam`.**class** `cdl.param.RadialProfileParam`

Paramètres de profil radial

**center**

Position du centre. Sélection unique parmi : “centroid”, “center”, “user”. Par défaut : “centroid”.

**Type**`guidata.dataset.dataitems.ChoiceItem`**x0** $X_{\text{Centre}}$ . Flottant, unité : pixel. Par défaut : None.**Type**`guidata.dataset.dataitems.FloatItem`**y0** $X_{\text{Centre}}$ . Flottant, unité : pixel. Par défaut : None.**Type**`guidata.dataset.dataitems.FloatItem`

**classmethod create**(*center* : *str*, *x0* : *float*, *y0* : *float*) → `cdl.computation.image.RadialProfileParam`

Renvoie une nouvelle instance de `RadialProfileParam` avec les champs initialisés aux valeurs données.**Paramètres**

- **center** (*str*) – Position du centre. Sélection unique parmi : “centroid”, “center”, “user”. Par défaut : “centroid”.
- **x0** (*float*) –  $X_{\text{Centre}}$ . Flottant, unité : pixel. Par défaut : None.
- **y0** (*float*) –  $X_{\text{Centre}}$ . Flottant, unité : pixel. Par défaut : None.

**Renvoie**Nouvelle instance de `RadialProfileParam`.

**update\_from\_image**(*obj* : `ImageObj`) → `None`

Mettre à jour les paramètres à partir de l’image

**choice\_callback**(*item*, *value*)

Callback pour l’item de choix

**class** `cdl.param.ResizeParam`

Paramètres de redimensionnement

**zoom**

Par défaut : None.

**Type**`guidata.dataset.dataitems.FloatItem`**mode**

Sélection unique parmi : “constant”, “nearest”, “reflect”, “wrap”. Par défaut : “constant”.

**Type**`guidata.dataset.dataitems.ChoiceItem`**cval**Valeur utilisée pour les points situés en dehors des frontières de l’image d’origine (si le mode est “constant”).  
Par défaut : 0.0.**Type**`guidata.dataset.dataitems.FloatItem`**prefilter**

Par défaut : True.

**Type**`guidata.dataset.dataitems.BoolItem`**order**

Ordre. Ordre de l’interpolation de type spline. Entier compris entre 0 et 5. Par défaut : 3.

**Type**`guidata.dataset.dataitems.IntItem`**classmethod** `create`(*zoom* : *float*, *mode* : *str*, *cval* : *float*, *prefilter* : *bool*, *order* : *int*) →`cdl.computation.image.ResizeParam`Renvoie une nouvelle instance de `ResizeParam` avec les champs initialisés aux valeurs données.**Paramètres**

- **zoom** (*float*) – Par défaut : None.
- **mode** (*str*) – Sélection unique parmi : “constant”, “nearest”, “reflect”, “wrap”. Par défaut : “constant”.
- **cval** (*float*) – Valeur utilisée pour les points situés en dehors des frontières de l’image d’origine (si le mode est “constant”). Par défaut : 0.0.
- **prefilter** (*bool*) – Par défaut : True.
- **order** (*int*) – Ordre. Ordre de l’interpolation de type spline. Entier compris entre 0 et 5. Par défaut : 3.

**Renvoie**Nouvelle instance de `ResizeParam`.**class** `cdl.param.RotateParam`

Paramètres de rotation

**angle**

Angle (°). Par défaut : None.

**Type**`guidata.dataset.dataitems.FloatItem`**mode**

Sélection unique parmi : “constant”, “nearest”, “reflect”, “wrap”. Par défaut : “constant”.

**Type**`guidata.dataset.dataitems.ChoiceItem`**cval**

Valeur utilisée pour les points situés en dehors des frontières de l'image d'origine (si le mode est "constant").  
Par défaut : 0.0.

**Type**`guidata.dataset.dataitems.FloatItem`**reshape**

Redimensionner l'image de destination de sorte qu'elle puisse contenir la totalité de l'image source. Par défaut : False.

**Type**`guidata.dataset.dataitems.BoolItem`**prefilter**

Par défaut : True.

**Type**`guidata.dataset.dataitems.BoolItem`**order**

Ordre. Ordre de l'interpolation de type spline. Entier compris entre 0 et 5. Par défaut : 3.

**Type**`guidata.dataset.dataitems.IntItem`

**classmethod create**(*angle : float, mode : str, cval : float, reshape : bool, prefilter : bool, order : int*) → *cdl.computation.image.RotateParam*

Renvoie une nouvelle instance de `RotateParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **angle** (*float*) – Angle (°). Par défaut : None.
- **mode** (*str*) – Sélection unique parmi : "constant", "nearest", "reflect", "wrap". Par défaut : "constant".
- **cval** (*float*) – Valeur utilisée pour les points situés en dehors des frontières de l'image d'origine (si le mode est "constant"). Par défaut : 0.0.
- **reshape** (*bool*) – Redimensionner l'image de destination de sorte qu'elle puisse contenir la totalité de l'image source. Par défaut : False.
- **prefilter** (*bool*) – Par défaut : True.
- **order** (*int*) – Ordre. Ordre de l'interpolation de type spline. Entier compris entre 0 et 5. Par défaut : 3.

**Renvoie**

Nouvelle instance de `RotateParam`.

**class** `cdl.param.ZCalibrateParam`

Paramètres de calibration linéaire de l'image

**a**

Par défaut : 1.0.

**Type**`guidata.dataset.dataitems.FloatItem`**b**

Par défaut : 0.0.

**Type**`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(a : float, b : float) → cdl.computation.image.ZCalibrateParam`

Renvoie une nouvelle instance de `ZCalibrateParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **a** (*float*) – Par défaut : 1.0.
- **b** (*float*) – Par défaut : 0.0.

**Renvoie**

Nouvelle instance de `ZCalibrateParam`.

## Paramètres de seuillage

**class** `cdl.param.ThresholdParam`

Histogram threshold parameters

**method**

Méthode de seuillage. Sélection unique parmi : “manual”, “isodata”, “li”, “mean”, “minimum”, “otsu”, “triangle”, “yen”. Par défaut : “manual”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**bins**

Nombre de classes. Entier supérieur à 1. Par défaut : 256.

**Type**

`guidata.dataset.dataitems.IntItem`

**value**

Valeur de seuil. Par défaut : 0.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**operation**

Opération. Sélection unique parmi : “>”, “<”. Par défaut : “>”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**classmethod** `create(method : str, bins : int, value : float, operation : str) → cdl.computation.image.threshold.ThresholdParam`

Renvoie une nouvelle instance de `ThresholdParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **method** (*str*) – Méthode de seuillage. Sélection unique parmi : “manual”, “isodata”, “li”, “mean”, “minimum”, “otsu”, “triangle”, “yen”. Par défaut : “manual”.
- **bins** (*int*) – Nombre de classes. Entier supérieur à 1. Par défaut : 256.
- **value** (*float*) – Valeur de seuil. Par défaut : 0.0.
- **operation** (*str*) – Opération. Sélection unique parmi : “>”, “<”. Par défaut : “>”.

**Renvoie**

Nouvelle instance de `ThresholdParam`.

## Paramètres de correction d'exposition

### **class** `cdl.param.AdjustGammaParam`

Paramètres d'ajustement gamma

#### **gamma**

Facteur de correction gamma (plus la valeur est élevée, plus le contraste est important). Flottant supérieur à 0.0. Par défaut : 1.0.

#### **Type**

`guidata.dataset.dataitems.FloatItem`

#### **gain**

Facteur de gain (plus la valeur est élevée, plus le contraste est important). Flottant supérieur à 0.0. Par défaut : 1.0.

#### **Type**

`guidata.dataset.dataitems.FloatItem`

### **classmethod** `create(gamma : float, gain : float) → cdl.computation.image.exposure.AdjustGammaParam`

Renvoie une nouvelle instance de `AdjustGammaParam` avec les champs initialisés aux valeurs données.

#### **Paramètres**

- **gamma** (*float*) – Facteur de correction gamma (plus la valeur est élevée, plus le contraste est important). Flottant supérieur à 0.0. Par défaut : 1.0.
- **gain** (*float*) – Facteur de gain (plus la valeur est élevée, plus le contraste est important). Flottant supérieur à 0.0. Par défaut : 1.0.

#### **Renvoie**

Nouvelle instance de `AdjustGammaParam`.

### **class** `cdl.param.AdjustLogParam`

Paramètres d'ajustement logarithmique

#### **gain**

Facteur de gain (plus la valeur est élevée, plus le contraste est important). Flottant supérieur à 0.0. Par défaut : 1.0.

#### **Type**

`guidata.dataset.dataitems.FloatItem`

#### **inv**

Si vrai, appliquer une transformation logarithmique inverse. Par défaut : False.

#### **Type**

`guidata.dataset.dataitems.BoolItem`

### **classmethod** `create(gain : float, inv : bool) → cdl.computation.image.exposure.AdjustLogParam`

Renvoie une nouvelle instance de `AdjustLogParam` avec les champs initialisés aux valeurs données.

#### **Paramètres**

- **gain** (*float*) – Facteur de gain (plus la valeur est élevée, plus le contraste est important). Flottant supérieur à 0.0. Par défaut : 1.0.
- **inv** (*bool*) – Si vrai, appliquer une transformation logarithmique inverse. Par défaut : False.

#### **Renvoie**

Nouvelle instance de `AdjustLogParam`.

### **class** `cdl.param.AdjustSigmoidParam`

Paramètres d'ajustement sigmoïde

**cutoff**

Valeur de coupure (plus la valeur est élevée, plus le contraste est important). Flottant compris entre 0.0 et 1.0. Par défaut : 0.5.

**Type**

`guidata.dataset.dataitems.FloatItem`

**gain**

Facteur de gain (plus la valeur est élevée, plus le contraste est important). Flottant supérieur à 0.0. Par défaut : 10.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**inv**

Si vrai, appliquer une transformation sigmoïde inverse. Par défaut : False.

**Type**

`guidata.dataset.dataitems.BoolItem`

**classmethod** `create(cutoff : float, gain : float, inv : bool) →`

`cdl.computation.image.exposure.AdjustSigmoidParam`

Renvoie une nouvelle instance de `AdjustSigmoidParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **cutoff** (*float*) – Valeur de coupure (plus la valeur est élevée, plus le contraste est important). Flottant compris entre 0.0 et 1.0. Par défaut : 0.5.
- **gain** (*float*) – Facteur de gain (plus la valeur est élevée, plus le contraste est important). Flottant supérieur à 0.0. Par défaut : 10.0.
- **inv** (*bool*) – Si vrai, appliquer une transformation sigmoïde inverse. Par défaut : False.

**Renvoie**

Nouvelle instance de `AdjustSigmoidParam`.

**class** `cdl.param.EqualizeAdaptHistParam`

Paramètres d'égalisation d'histogramme adaptatif

**nbins**

Nombre de classes. Nombre de classes de l'histogramme des niveaux de l'image. Entier supérieur à 1. Par défaut : 256.

**Type**

`guidata.dataset.dataitems.IntItem`

**clip\_limit**

Écrêtage limite. Valeur d'écrtage (plus la valeur est élevée, plus le contraste est important). Flottant compris entre 0.0 et 1.0. Par défaut : 0.01.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(nbins : int, clip_limit : float) →`

`cdl.computation.image.exposure.EqualizeAdaptHistParam`

Renvoie une nouvelle instance de `EqualizeAdaptHistParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **nbins** (*int*) – Nombre de classes. Nombre de classes de l'histogramme des niveaux de l'image. Entier supérieur à 1. Par défaut : 256.
- **clip\_limit** (*float*) – Écrêtage limite. Valeur d'écrtage (plus la valeur est élevée, plus le contraste est important). Flottant compris entre 0.0 et 1.0. Par défaut : 0.01.

**Renvoie**

Nouvelle instance de `EqualizeAdaptHistParam`.

**class** `cdl.param.EqualizeHistParam`

Paramètres d'égalisation d'histogramme

**nbins**

Nombre de classes. Nombre de classes de l'histogramme des niveaux de l'image. Entier supérieur à 1. Par défaut : 256.

**Type**

`guidata.dataset.dataitems.IntItem`

**classmethod** `create(nbins : int) → cdl.computation.image.exposure.EqualizeHistParam`

Renvoie une nouvelle instance de `EqualizeHistParam` avec les champs initialisés aux valeurs données.

**Paramètres**

**nbins** (*int*) – Nombre de classes. Nombre de classes de l'histogramme des niveaux de l'image. Entier supérieur à 1. Par défaut : 256.

**Renvoie**

Nouvelle instance de `EqualizeHistParam`.

**class** `cdl.param.RescaleIntensityParam`

Paramètres de rééchantillonnage d'intensité

**in\_range**

Echelle de niveaux en entrée. Valeurs min/max d'intensité de l'image en entrée ("image" correspond aux niveaux min/max de l'image en entrée, "dtype" correspond aux valeurs min/max du type de données de l'image). Sélection unique parmi : "image", "dtype", "float32", "float64", "complex128", "int32", "int16", "uint16", "uint8". Par défaut : "image".

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**out\_range**

Echelle de niveaux en sortie. Valeurs min/max d'intensité de l'image en sortie ("image" correspond aux niveaux min/max de l'image en entrée, "dtype" correspond aux valeurs min/max du type de données de l'image). Sélection unique parmi : "image", "dtype", "float32", "float64", "complex128", "int32", "int16", "uint16", "uint8". Par défaut : "dtype".

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**classmethod** `create(in_range : str, out_range : str) →`

`cdl.computation.image.exposure.RescaleIntensityParam`

Renvoie une nouvelle instance de `RescaleIntensityParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **in\_range** (*str*) – Echelle de niveaux en entrée. Valeurs min/max d'intensité de l'image en entrée ("image" correspond aux niveaux min/max de l'image en entrée, "dtype" correspond aux valeurs min/max du type de données de l'image). Sélection unique parmi : "image", "dtype", "float32", "float64", "complex128", "int32", "int16", "uint16", "uint8". Par défaut : "image".
- **out\_range** (*str*) – Echelle de niveaux en sortie. Valeurs min/max d'intensité de l'image en sortie ("image" correspond aux niveaux min/max de l'image en entrée, "dtype" correspond aux valeurs min/max du type de données de l'image). Sélection unique parmi : "image", "dtype", "float32", "float64", "complex128", "int32", "int16", "uint16", "uint8". Par défaut : "dtype".

**Renvoie**

Nouvelle instance de `RescaleIntensityParam`.

**Paramètres de restauration****class** `cdl.param.DenoiseBilateralParam`

Paramètres de débruitage par filtre bilatéral

**sigma\_spatial**

spatial. Ecart-type dans le domaine spatial. Une valeur élevée a pour effet de moyenner des pixels séparés par de grandes distances. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**mode**

Sélection unique parmi : “constant”, “edge”, “symmetric”, “reflect”, “wrap”. Par défaut : “constant”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**cval**

Valeur de remplissage en dehors des bornes de l’image (en mode constant). Par défaut : 0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(sigma_spatial : float, mode : str, cval : float) →`

`cdl.computation.image.restoration.DenoiseBilateralParam`

Renvoie une nouvelle instance de `DenoiseBilateralParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **sigma\_spatial** (*float*) – spatial. Ecart-type dans le domaine spatial. Une valeur élevée a pour effet de moyenner des pixels séparés par de grandes distances. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 1.0.
- **mode** (*str*) – Sélection unique parmi : “constant”, “edge”, “symmetric”, “reflect”, “wrap”. Par défaut : “constant”.
- **cval** (*float*) – Valeur de remplissage en dehors des bornes de l’image (en mode constant). Par défaut : 0.

**Renvoie**

Nouvelle instance de `DenoiseBilateralParam`.

**class** `cdl.param.DenoiseTVParam`

Paramètres de débruitage par variation totale

**weight**

Poids de débruitage. Plus le poids est élevé, plus le débruitage est fort (aux dépens de la fidélité des données). Flottant supérieur à 0, non nul. Par défaut : 0.1.

**Type**

`guidata.dataset.dataitems.FloatItem`

**eps**

Epsilon. Différence relative de la valeur de la fonction de coût qui détermine le critère d’arrêt de l’algorithme. Ce dernier s’arrête lorsque :  $(e_{(n-1)} - e_n) < \text{eps} * e_0$ . Flottant supérieur à 0, non nul. Par défaut : 0.0002.

**Type**

`guidata.dataset.dataitems.FloatItem`

**max\_num\_iter**

Nb. max. d'itérations. Nombre maximal d'itérations utilisé pour l'optimisation. Entier supérieur à 0, non nul. Par défaut : 200.

**Type**

`guidata.dataset.dataitems.IntItem`

**classmethod** `create(weight : float, eps : float, max_num_iter : int) →`  
`cdl.computation.image.restoration.DenoiseTVParam`

Renvoie une nouvelle instance de `DenoiseTVParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **weight** (*float*) – Poids de débruitage. Plus le poids est élevé, plus le débruitage est fort (aux dépens de la fidélité des données). Flottant supérieur à 0, non nul. Par défaut : 0.1.
- **eps** (*float*) – Epsilon. Différence relative de la valeur de la fonction de coût qui détermine le critère d'arrêt de l'algorithme. Ce dernier s'arrête lorsque :  $(e_{(n-1)} - e_n) < \text{eps} * e_0$ . Flottant supérieur à 0, non nul. Par défaut : 0.0002.
- **max\_num\_iter** (*int*) – Nb. max. d'itérations. Nombre maximal d'itérations utilisé pour l'optimisation. Entier supérieur à 0, non nul. Par défaut : 200.

**Renvoie**

Nouvelle instance de `DenoiseTVParam`.

**class** `cdl.param.DenoiseWaveletParam`

Paramètres de débruitage par ondelettes

**wavelet**

Ondelette. Sélection unique parmi : "bior1.1", "bior1.3", "bior1.5", "bior2.2", "bior2.4", "bior2.6", "bior2.8", "bior3.1", "bior3.3", "bior3.5", "bior3.7", "bior3.9", "bior4.4", "bior5.5", "bior6.8", "cgau1", "cgau2", "cgau3", "cgau4", "cgau5", "cgau6", "cgau7", "cgau8", "cmor", "coif1", "coif2", "coif3", "coif4", "coif5", "coif6", "coif7", "coif8", "coif9", "coif10", "coif11", "coif12", "coif13", "coif14", "coif15", "coif16", "coif17", "db1", "db2", "db3", "db4", "db5", "db6", "db7", "db8", "db9", "db10", "db11", "db12", "db13", "db14", "db15", "db16", "db17", "db18", "db19", "db20", "db21", "db22", "db23", "db24", "db25", "db26", "db27", "db28", "db29", "db30", "db31", "db32", "db33", "db34", "db35", "db36", "db37", "db38", "dmey", "fbsp", "gaus1", "gaus2", "gaus3", "gaus4", "gaus5", "gaus6", "gaus7", "gaus8", "haar", "mexh", "morl", "rbio1.1", "rbio1.3", "rbio1.5", "rbio2.2", "rbio2.4", "rbio2.6", "rbio2.8", "rbio3.1", "rbio3.3", "rbio3.5", "rbio3.7", "rbio3.9", "rbio4.4", "rbio5.5", "rbio6.8", "shan", "sym2", "sym3", "sym4", "sym5", "sym6", "sym7", "sym8", "sym9", "sym10", "sym11", "sym12", "sym13", "sym14", "sym15", "sym16", "sym17", "sym18", "sym19", "sym20". Par défaut : "sym9".

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**mode**

Sélection unique parmi : "soft", "hard". Par défaut : "soft".

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**method**

Méthode. Sélection unique parmi : "BayesShrink", "VisuShrink". Par défaut : "VisuShrink".

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**classmethod** `create(wavelet : str, mode : str, method : str) →`  
`cdl.computation.image.restoration.DenoiseWaveletParam`

Renvoie une nouvelle instance de `DenoiseWaveletParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **wavelet** (*str*) – Ondelette. Sélection unique parmi : “bior1.1”, “bior1.3”, “bior1.5”, “bior2.2”, “bior2.4”, “bior2.6”, “bior2.8”, “bior3.1”, “bior3.3”, “bior3.5”, “bior3.7”, “bior3.9”, “bior4.4”, “bior5.5”, “bior6.8”, “cgau1”, “cgau2”, “cgau3”, “cgau4”, “cgau5”, “cgau6”, “cgau7”, “cgau8”, “cmor”, “coif1”, “coif2”, “coif3”, “coif4”, “coif5”, “coif6”, “coif7”, “coif8”, “coif9”, “coif10”, “coif11”, “coif12”, “coif13”, “coif14”, “coif15”, “coif16”, “coif17”, “db1”, “db2”, “db3”, “db4”, “db5”, “db6”, “db7”, “db8”, “db9”, “db10”, “db11”, “db12”, “db13”, “db14”, “db15”, “db16”, “db17”, “db18”, “db19”, “db20”, “db21”, “db22”, “db23”, “db24”, “db25”, “db26”, “db27”, “db28”, “db29”, “db30”, “db31”, “db32”, “db33”, “db34”, “db35”, “db36”, “db37”, “db38”, “dmey”, “fbsp”, “gaus1”, “gaus2”, “gaus3”, “gaus4”, “gaus5”, “gaus6”, “gaus7”, “gaus8”, “haar”, “mexh”, “morl”, “rbio1.1”, “rbio1.3”, “rbio1.5”, “rbio2.2”, “rbio2.4”, “rbio2.6”, “rbio2.8”, “rbio3.1”, “rbio3.3”, “rbio3.5”, “rbio3.7”, “rbio3.9”, “rbio4.4”, “rbio5.5”, “rbio6.8”, “shan”, “sym2”, “sym3”, “sym4”, “sym5”, “sym6”, “sym7”, “sym8”, “sym9”, “sym10”, “sym11”, “sym12”, “sym13”, “sym14”, “sym15”, “sym16”, “sym17”, “sym18”, “sym19”, “sym20”. Par défaut : “sym9”.
- **mode** (*str*) – Sélection unique parmi : “soft”, “hard”. Par défaut : “soft”.
- **method** (*str*) – Méthode. Sélection unique parmi : “BayesShrink”, “VisuShrink”. Par défaut : “VisuShrink”.

**Renvoie**

Nouvelle instance de `DenoiseWaveletParam`.

**Paramètres morphologiques**

**class** `cdl.param.MorphologyParam`

Paramètres de White Top-Hat

**radius**

Rayon. Rayon de l'ouverture circulaire (disque). Entier supérieur à 1. Par défaut : 1.

**Type**

`guidata.dataset.dataitems.IntItem`

**classmethod** `create(radius : int) → cdl.computation.image.morphology.MorphologyParam`

Renvoie une nouvelle instance de `MorphologyParam` avec les champs initialisés aux valeurs données.

**Paramètres**

**radius** (*int*) – Rayon. Rayon de l'ouverture circulaire (disque). Entier supérieur à 1. Par défaut : 1.

**Renvoie**

Nouvelle instance de `MorphologyParam`.

**Paramètres de détection de contours**

**class** `cdl.param.CannyParam`

Paramètres de filtre Canny

**sigma**

Ecart-type du filtrage gaussien. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**low\_threshold**

Seuil bas. Borne inférieure pour le seuillage par hystérésis (liaison des contours). Flottant supérieur à 0. Par défaut : 0.1.

**Type**`guidata.dataset.dataitems.FloatItem`**high\_threshold**

Seuil haut. Borne supérieure pour le seuillage par hystérésis (liaison des contours). Flottant supérieur à 0. Par défaut : 0.9.

**Type**`guidata.dataset.dataitems.FloatItem`**use\_quantiles**

Interprète les seuils bas et haut en tant que quantiles des niveaux des contours, au lieu de valeurs absolues des contours. Si le réglage est actif, alors les seuils doivent être compris entre 0 et 1. Par défaut : True.

**Type**`guidata.dataset.dataitems.BoolItem`**mode**

Sélection unique parmi : “reflect”, “constant”, “nearest”, “mirror”, “wrap”. Par défaut : “constant”.

**Type**`guidata.dataset.dataitems.ChoiceItem`**cval**

Valeur de remplissage si le mode est constant. Par défaut : 0.0.

**Type**`guidata.dataset.dataitems.FloatItem`

**classmethod create**(*sigma : float, low\_threshold : float, high\_threshold : float, use\_quantiles : bool, mode : str, cval : float*) → *cdl.computation.image.edges.CannyParam*

Renvoie une nouvelle instance de `CannyParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **sigma** (*float*) – Ecart-type du filtrage gaussien. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 1.0.
- **low\_threshold** (*float*) – Seuil bas. Borne inférieure pour le seuillage par hystérésis (liaison des contours). Flottant supérieur à 0. Par défaut : 0.1.
- **high\_threshold** (*float*) – Seuil haut. Borne supérieure pour le seuillage par hystérésis (liaison des contours). Flottant supérieur à 0. Par défaut : 0.9.
- **use\_quantiles** (*bool*) – Interprète les seuils bas et haut en tant que quantiles des niveaux des contours, au lieu de valeurs absolues des contours. Si le réglage est actif, alors les seuils doivent être compris entre 0 et 1. Par défaut : True.
- **mode** (*str*) – Sélection unique parmi : “reflect”, “constant”, “nearest”, “mirror”, “wrap”. Par défaut : “constant”.
- **cval** (*float*) – Valeur de remplissage si le mode est constant. Par défaut : 0.0.

**Renvoie**

Nouvelle instance de `CannyParam`.

**Paramètres de détection****class cdl.param.BlobDOGParam**

Détection de taches par méthode de différence de gaussiennes

**min\_sigma**

min. L'écart-type minimal pour le noyau gaussien. Cette valeur doit être faible pour détecter de petites taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 1.0.

**Type**`guidata.dataset.dataitems.FloatItem`**max\_sigma**

max. L'écart-type maximal pour le noyau gaussien. Cette valeur doit être élevée pour détecter de grandes taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 30.0.

**Type**`guidata.dataset.dataitems.FloatItem`**threshold\_rel**

Seuil relatif. Intensité minimale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.2.

**Type**`guidata.dataset.dataitems.FloatItem`**overlap**

Recouvrement. Si deux taches ont un taux de recouvrement supérieur à ce seuil, alors la plus petite tache est éliminée. Flottant compris entre 0.0 et 1.0. Par défaut : 0.5.

**Type**`guidata.dataset.dataitems.FloatItem`**exclude\_border**

Si le réglage est actif, exclure les taches de la bordure de l'image. Par défaut : True.

**Type**`guidata.dataset.dataitems.BoolItem`

**classmethod create**(*min\_sigma : float, max\_sigma : float, threshold\_rel : float, overlap : float, exclude\_border : bool*) → *cdl.computation.image.detection.BlobDOGParam*

Renvoie une nouvelle instance de `BlobDOGParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **min\_sigma** (*float*) – min. L'écart-type minimal pour le noyau gaussien. Cette valeur doit être faible pour détecter de petites taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 1.0.
- **max\_sigma** (*float*) – max. L'écart-type maximal pour le noyau gaussien. Cette valeur doit être élevée pour détecter de grandes taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 30.0.
- **threshold\_rel** (*float*) – Seuil relatif. Intensité minimale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.2.
- **overlap** (*float*) – Recouvrement. Si deux taches ont un taux de recouvrement supérieur à ce seuil, alors la plus petite tache est éliminée. Flottant compris entre 0.0 et 1.0. Par défaut : 0.5.
- **exclude\_border** (*bool*) – Si le réglage est actif, exclure les taches de la bordure de l'image. Par défaut : True.

**Renvoie**

Nouvelle instance de `BlobDOGParam`.

**class** `cdl.param.BlobDOHParam`

Détection de taches par méthode du déterminant du Hessian

**min\_sigma**

min. L'écart-type minimal pour le noyau gaussien. Cette valeur doit être faible pour détecter de petites taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 1.0.

**Type**`guidata.dataset.dataitems.FloatItem`

**max\_sigma**

max. L'écart-type maximal pour le noyau gaussien. Cette valeur doit être élevée pour détecter de grandes taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 30.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**threshold\_rel**

Seuil relatif. Intensité minimale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.2.

**Type**

`guidata.dataset.dataitems.FloatItem`

**overlap**

Recouvrement. Si deux taches ont un taux de recouvrement supérieur à ce seuil, alors la plus petite tache est éliminée. Flottant compris entre 0.0 et 1.0. Par défaut : 0.5.

**Type**

`guidata.dataset.dataitems.FloatItem`

**log\_scale**

Les valeurs intermédiaires d'écart-type peuvent être interpolées selon une échelle linéaire ou logarithmique. Par défaut : False.

**Type**

`guidata.dataset.dataitems.BoolItem`

**classmethod create**(*min\_sigma : float, max\_sigma : float, threshold\_rel : float, overlap : float, log\_scale : bool*) → `cdl.computation.image.detection.BlobDOHParam`

Renvoie une nouvelle instance de `BlobDOHParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **min\_sigma** (*float*) – min. L'écart-type minimal pour le noyau gaussien. Cette valeur doit être faible pour détecter de petites taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 1.0.
- **max\_sigma** (*float*) – max. L'écart-type maximal pour le noyau gaussien. Cette valeur doit être élevée pour détecter de grandes taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 30.0.
- **threshold\_rel** (*float*) – Seuil relatif. Intensité minimale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.2.
- **overlap** (*float*) – Recouvrement. Si deux taches ont un taux de recouvrement supérieur à ce seuil, alors la plus petite tache est éliminée. Flottant compris entre 0.0 et 1.0. Par défaut : 0.5.
- **log\_scale** (*bool*) – Les valeurs intermédiaires d'écart-type peuvent être interpolées selon une échelle linéaire ou logarithmique. Par défaut : False.

**Renvoie**

Nouvelle instance de `BlobDOHParam`.

**class** `cdl.param.BlobLOGParam`

Détection de taches par méthode du Laplacien de gaussiennes

**min\_sigma**

min. L'écart-type minimal pour le noyau gaussien. Cette valeur doit être faible pour détecter de petites taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**max\_sigma**

max. L'écart-type maximal pour le noyau gaussien. Cette valeur doit être élevée pour détecter de grandes taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 30.0.

**Type**`guidata.dataset.dataitems.FloatItem`**threshold\_rel**

Seuil relatif. Intensité minimale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.2.

**Type**`guidata.dataset.dataitems.FloatItem`**overlap**

Recouvrement. Si deux taches ont un taux de recouvrement supérieur à ce seuil, alors la plus petite tache est éliminée. Flottant compris entre 0.0 et 1.0. Par défaut : 0.5.

**Type**`guidata.dataset.dataitems.FloatItem`**log\_scale**

Les valeurs intermédiaires d'écart-type peuvent être interpolées selon une échelle linéaire ou logarithmique. Par défaut : False.

**Type**`guidata.dataset.dataitems.BoolItem`**exclude\_border**

Si le réglage est actif, exclure les taches de la bordure de l'image. Par défaut : True.

**Type**`guidata.dataset.dataitems.BoolItem`

**classmethod create**(*min\_sigma : float, max\_sigma : float, threshold\_rel : float, overlap : float, log\_scale : bool, exclude\_border : bool*) → `cdl.computation.image.detection.BlobLOGParam`

Renvoie une nouvelle instance de `BlobLOGParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **min\_sigma** (*float*) – <sub>min</sub>. L'écart-type minimal pour le noyau gaussien. Cette valeur doit être faible pour détecter de petites taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 1.0.
- **max\_sigma** (*float*) – <sub>max</sub>. L'écart-type maximal pour le noyau gaussien. Cette valeur doit être élevée pour détecter de grandes taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 30.0.
- **threshold\_rel** (*float*) – Seuil relatif. Intensité minimale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.2.
- **overlap** (*float*) – Recouvrement. Si deux taches ont un taux de recouvrement supérieur à ce seuil, alors la plus petite tache est éliminée. Flottant compris entre 0.0 et 1.0. Par défaut : 0.5.
- **log\_scale** (*bool*) – Les valeurs intermédiaires d'écart-type peuvent être interpolées selon une échelle linéaire ou logarithmique. Par défaut : False.
- **exclude\_border** (*bool*) – Si le réglage est actif, exclure les taches de la bordure de l'image. Par défaut : True.

**Renvoie**

Nouvelle instance de `BlobLOGParam`.

**class** `cdl.param.BlobOpenCVParam`

Détection de taches par OpenCV

**min\_threshold**

Seuil min. Le seuil minimum entre les maxima et minima locaux. Ce paramètre n'affecte pas la qualité des taches, mais seulement leur nombre. Un seuil bas donne un nombre plus important de taches. Flottant supérieur à 0.0. Par défaut : 10.0.

**Type**`guidata.dataset.dataitems.FloatItem`**max\_threshold**

Seuil max. Le seuil maximum entre les maxima et minima locaux. Ce paramètre n'affecte pas la qualité des taches, mais seulement leur nombre. Un seuil bas donne un nombre plus important de taches. Flottant supérieur à 0.0. Par défaut : 200.0.

**Type**`guidata.dataset.dataitems.FloatItem`**min\_repeatability**

Répétabilité min. Le nombre minimum de fois qu'une tache doit être détectée dans une séquence d'images pour être considérée valide. Entier supérieur à 1. Par défaut : 2.

**Type**`guidata.dataset.dataitems.IntItem`**min\_dist\_between\_blobs**

Distance min. entre taches. La distance minimale entre deux taches. Si deux taches sont détectées à une distance inférieure à celle-ci, alors la plus petite tache est éliminée. Flottant supérieur à 0.0, non nul. Par défaut : 10.0.

**Type**`guidata.dataset.dataitems.FloatItem`**filter\_by\_color**

Si vrai, l'image est filtrée par couleur au lieu d'intensité. Par défaut : True.

**Type**`guidata.dataset.dataitems.BoolItem`**blob\_color**

Couleur de la tache. La couleur des taches à détecter (0 pour les taches foncées, 255 pour les taches claires). Par défaut : 0.

**Type**`guidata.dataset.dataitems.IntItem`**filter\_by\_area**

Si vrai, l'image est filtrée par l'aire des taches. Par défaut : True.

**Type**`guidata.dataset.dataitems.BoolItem`**min\_area**

Aire min. L'aire minimale des taches. Flottant supérieur à 0.0. Par défaut : 25.0.

**Type**`guidata.dataset.dataitems.FloatItem`**max\_area**

Aire max. L'aire maximale des taches. Flottant supérieur à 0.0. Par défaut : 500.0.

**Type**`guidata.dataset.dataitems.FloatItem`**filter\_by\_circularity**

Si vrai, l'image est filtrée par la circularité des taches. Par défaut : False.

**Type**`guidata.dataset.dataitems.BoolItem`

**min\_circularity**

Circularité min. La circularité minimale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.8.

**Type**

`guidata.dataset.dataitems.FloatItem`

**max\_circularity**

Circularité max. La circularité maximale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**filter\_by\_inertia**

Si vrai, l'image est filtrée par l'inertie des taches. Par défaut : False.

**Type**

`guidata.dataset.dataitems.BoolItem`

**min\_inertia\_ratio**

Ratio d'inertie min. Le ratio d'inertie minimal des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.6.

**Type**

`guidata.dataset.dataitems.FloatItem`

**max\_inertia\_ratio**

Ratio d'inertie max. Le ratio d'inertie maximal des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**filter\_by\_convexity**

Si vrai, l'image est filtrée par la convexité des taches. Par défaut : False.

**Type**

`guidata.dataset.dataitems.BoolItem`

**min\_convexity**

Convexité min. La convexité minimale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.8.

**Type**

`guidata.dataset.dataitems.FloatItem`

**max\_convexity**

Convexité max. La convexité maximale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod create**(*min\_threshold : float, max\_threshold : float, min\_repeatability : int, min\_dist\_between\_blobs : float, filter\_by\_color : bool, blob\_color : int, filter\_by\_area : bool, min\_area : float, max\_area : float, filter\_by\_circularity : bool, min\_circularity : float, max\_circularity : float, filter\_by\_inertia : bool, min\_inertia\_ratio : float, max\_inertia\_ratio : float, filter\_by\_convexity : bool, min\_convexity : float, max\_convexity : float*) → *cdl.computation.image.detection.BlobOpenCVParam*

Renvoie une nouvelle instance de `BlobOpenCVParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **min\_threshold** (*float*) – Seuil min. Le seuil minimum entre les maxima et minima locaux. Ce paramètre n'affecte pas la qualité des taches, mais seulement leur nombre. Un seuil bas donne un nombre plus important de taches. Flottant supérieur à 0.0. Par défaut : 10.0.

- **max\_threshold** (*float*) – Seuil max. Le seuil maximum entre les maxima et minima locaux. Ce paramètre n’affecte pas la qualité des taches, mais seulement leur nombre. Un seuil bas donne un nombre plus important de taches. Flottant supérieur à 0.0. Par défaut : 200.0.
- **min\_repeatability** (*int*) – Répétabilité min. Le nombre minimum de fois qu’une tache doit être détectée dans une séquence d’images pour être considérée valide. Entier supérieur à 1. Par défaut : 2.
- **min\_dist\_between\_blobs** (*float*) – Distance min. entre taches. La distance minimale entre deux taches. Si deux taches sont détectées à une distance inférieure à celle-ci, alors la plus petite tache est éliminée. Flottant supérieur à 0.0, non nul. Par défaut : 10.0.
- **filter\_by\_color** (*bool*) – Si vrai, l’image est filtrée par couleur au lieu d’intensité. Par défaut : True.
- **blob\_color** (*int*) – Couleur de la tache. La couleur des taches à détecter (0 pour les taches foncées, 255 pour les taches claires). Par défaut : 0.
- **filter\_by\_area** (*bool*) – Si vrai, l’image est filtrée par l’aire des taches. Par défaut : True.
- **min\_area** (*float*) – Aire min. L’aire minimale des taches. Flottant supérieur à 0.0. Par défaut : 25.0.
- **max\_area** (*float*) – Aire max. L’aire maximale des taches. Flottant supérieur à 0.0. Par défaut : 500.0.
- **filter\_by\_circularity** (*bool*) – Si vrai, l’image est filtrée par la circularité des taches. Par défaut : False.
- **min\_circularity** (*float*) – Circularité min. La circularité minimale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.8.
- **max\_circularity** (*float*) – Circularité max. La circularité maximale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 1.0.
- **filter\_by\_inertia** (*bool*) – Si vrai, l’image est filtrée par l’inertie des taches. Par défaut : False.
- **min\_inertia\_ratio** (*float*) – Ratio d’inertie min. Le ratio d’inertie minimal des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.6.
- **max\_inertia\_ratio** (*float*) – Ratio d’inertie max. Le ratio d’inertie maximal des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 1.0.
- **filter\_by\_convexity** (*bool*) – Si vrai, l’image est filtrée par la convexité des taches. Par défaut : False.
- **min\_convexity** (*float*) – Convexité min. La convexité minimale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.8.
- **max\_convexity** (*float*) – Convexité max. La convexité maximale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 1.0.

**Renvoie**

Nouvelle instance de BlobOpenCVPParam.

**class cdl.param.ContourShapeParam**

Paramètres de forme de contour

**threshold**

Seuil relatif. Seuil de détection, relatif à la différence entre le maximum et le minimum des données de l’image. Flottant compris entre 0.1 et 0.9. Par défaut : 0.5.

**Type**

`guidata.dataset.dataitems.FloatItem`

**shape**

Forme. Sélection unique parmi : “ellipse”, “circle”, “polygon”. Par défaut : “ellipse”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**classmethod create**(*threshold : float, shape : str*) →  
*cdl.computation.image.detection.ContourShapeParam*

Renvoie une nouvelle instance de `ContourShapeParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **threshold** (*float*) – Seuil relatif. Seuil de détection, relatif à la différence entre le maximum et le minimum des données de l’image. Flottant compris entre 0.1 et 0.9. Par défaut : 0.5.
- **shape** (*str*) – Forme. Sélection unique parmi : “ellipse”, “circle”, “polygon”. Par défaut : “ellipse”.

**Renvoie**

Nouvelle instance de `ContourShapeParam`.

**class** `cdl.param.Peak2DDetectionParam`

Paramètres de détection de pics

**threshold**

Seuil relatif. Seuil de détection, relatif à la différence entre le maximum et le minimum des données de l’image. Flottant compris entre 0.1 et 0.9. Par défaut : 0.5.

**Type**

*guidata.dataset.dataitems.FloatItem*

**size**

Taille du voisinage. Taille de la fenêtre glissante utilisée dans l’algorithme de filtrage maximum/minimum. Entier supérieur à 1, unité : pixels. Par défaut : 10.

**Type**

*guidata.dataset.dataitems.IntItem*

**create\_rois**

Par défaut : True.

**Type**

*guidata.dataset.dataitems.BoolItem*

**classmethod create**(*threshold : float, size : int, create\_rois : bool*) →  
*cdl.computation.image.detection.Peak2DDetectionParam*

Renvoie une nouvelle instance de `Peak2DDetectionParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **threshold** (*float*) – Seuil relatif. Seuil de détection, relatif à la différence entre le maximum et le minimum des données de l’image. Flottant compris entre 0.1 et 0.9. Par défaut : 0.5.
- **size** (*int*) – Taille du voisinage. Taille de la fenêtre glissante utilisée dans l’algorithme de filtrage maximum/minimum. Entier supérieur à 1, unité : pixels. Par défaut : 10.
- **create\_rois** (*bool*) – Par défaut : True.

**Renvoie**

Nouvelle instance de `Peak2DDetectionParam`.

### 3.3 Modèle de données (cdl.obj)

Le module `cdl.obj` fournit les classes et fonctions nécessaires pour créer et manipuler des objets signal et image.

Ces classes et fonctions sont définies dans d'autres modules :

- `cdl.core.model.base`
- `cdl.core.model.image`
- `cdl.core.model.signal`
- `cdl.core.io`

Le module `cdl.obj` est donc un moyen pratique d'importer tous les objets en une seule fois. En fait, l'instruction d'importation suivante est équivalente à la précédente :

```
# Original import statement
from cdl.core.model.signal import SignalObj
from cdl.core.model.image import ImageObj

# Equivalent import statement
from cdl.obj import SignalObj, ImageObj
```

#### 3.3.1 Objets communs

**class** `cdl.obj.ResultProperties`(*title* : *str*, *array* : *ndarray*, *labels* : *list[str] | None*, *item\_json* : *str* = "")

Object representing properties serializable in signal/image metadata.

Result *array* is a NumPy 2-D array : each row is a list of properties, optionnally associated to a ROI (first column value).

L'index de la ROI commence à 0 (ou est simplement 0 s'il n'y a pas de ROI).

##### Paramètres

- **title** – properties title
- **array** – properties array
- **labels** – properties labels (one label per column of result array)
- **item\_json** – JSON string of label item associated to this obj

**Note :** The *array* argument can be a list of lists or a NumPy array. For instance, the following are equivalent :

- `array = [[1, 2], [3, 4]]`
- `array = np.array([[1, 2], [3, 4]])`

Or for only one line (one single result), the following are equivalent :

- `array = [1, 2]`
- `array = [[1, 2]]`
- `array = np.array([[1, 2]])`

**property category:** *str*

Return result category

**property headers:** *list[str] | None*

Return result headers (one header per column of result array)

**property shown\_array:** *ndarray*

Return array of shown results, i.e. including complementary array (if any)

##### Renvoie

Tableau des résultats affichés

**update\_obj\_metadata\_from\_item**(*obj* : *BaseObj*, *item* : *LabelItem* | *None*) → *None*

Update object metadata with label item

**Paramètres**

- **obj** – object (signal/image)
- **item** – label item

**property label\_contents**: *tuple*[*tuple*[*int*, *str*], ...]

Return label contents, i.e. a tuple of couples of (index, text) where index is the column of raw\_data and text is the associated label format string

**create\_label\_item**(*obj* : *BaseObj*) → *LabelItem* | *None*

Create label item

**Paramètres**

- obj** – object (signal/image)

**Renvoie**

Label item

---

**Note :** The signal or image object is required as argument to create the label item because the label text may contain format strings that need to be filled with the object properties. For instance, the label text may contain the signal or image units.

---

**get\_label\_item**(*obj* : *BaseObj*) → *LabelItem* | *None*

Return label item associated to this result

**Paramètres**

- obj** – object (signal/image)

**Renvoie**

Label item

---

**Note :** The signal or image object is required as argument to eventually create the label item if it has not been created yet. See [create\\_label\\_item\(\)](#).

---

**class cdl.obj.ResultShape**(*title* : *str*, *array* : *ndarray*, *shape* : *Literall*['rectangle', 'circle', 'ellipse', 'segment', 'marker', 'point', 'polygon'], *item\_json* : *str* = "", *add\_label* : *bool* = *False*)

Objet représentant une forme géométrique sérialisable dans les métadonnées du signal/image.

Le tableau de résultats *array* est un tableau NumPy 2-D : chaque ligne est un résultat, éventuellement associé à une ROI (valeur de la première colonne).

L'index de la ROI commence à 0 (ou est simplement 0 s'il n'y a pas de ROI).

**Paramètres**

- **title** – result shape title
- **array** – coordonnées de la forme (plusieurs formes : une forme par ligne), la première colonne est l'index de la ROI (0 s'il n'y a pas de ROI)
- **shape** – shape kind
- **item\_json** – JSON string of label item associated to this obj
- **add\_label** – if True, add a label item (and the geometrical shape) to plot (default to False)

**Lève**

- AssertionError** – argument invalide

---

**Note :** The *array* argument can be a list of lists or a NumPy array. For instance, the following are equivalent :

— *array* = [[1, 2], [3, 4]]

```
— array = np.array([[1, 2], [3, 4]])  
Or for only one line (one single result), the following are equivalent :  
— array = [1, 2]  
— array = [[1, 2]]  
— array = np.array([[1, 2]])
```

---

**property category:** `str`

Return result category

**check\_array()** → `None`

Check if array attribute is valid

**Lève**

`AssertionError` – invalid array

**property headers:** `list[str] | None`

Return result headers (one header per column of result array)

**property shown\_array:** `ndarray`

Return array of shown results, i.e. including complementary array (if any)

**Renvoie**

Tableau des résultats affichés

**property label\_contents:** `tuple[tuple[int, str], ...]`

Return label contents, i.e. a tuple of couples of (index, text) where index is the column of raw\_data and text is the associated label format string

**create\_label\_item**(*obj* : *BaseObj*) → `LabelItem | None`

Create label item

**Renvoie**

Label item

**merge\_with**(*obj* : *BaseObj*, *other\_metadata* : *dict[str, Any]*) → `None`

Merge object resultshape with another's metadata (obj <- other obj's metadata)

**Paramètres**

- **obj** – object (signal/image)
- **other\_metadata** – other object metadata

**transform\_coordinates**(*func* : *Callable[[ndarray], None]*) → `None`

Transforme les coordonnées de la forme.

**Paramètres**

**func** – fonction pour transformer les coordonnées

**iterate\_plot\_items**(*fmt* : *str*, *lbl* : *bool*, *option* : *Literal['s', 'i']*) → `Iterable`

Itère sur les éléments de tracé de la forme de métadonnées.

**Paramètres**

- **fmt** – format numérique (par exemple « `%0.3f` »)
- **lbl** – si True, affiche les étiquettes de forme
- **option** – shape style option (« s » for signal, « i » for image)

**Produit** (*yield*)

Item graphique

**create\_shape\_item**(*coords* : *np.ndarray*, *fmt* : *str*, *lbl* : *bool*, *option* : *Literal['s', 'i']*) → `AnnotatedPoint | Marker | AnnotatedRectangle | AnnotatedCircle | AnnotatedSegment | AnnotatedEllipse | PolygonShape | None`

Make geometrical shape plot item adapted to the shape type.

**Paramètres**

- **coords** – données de forme
- **fmt** – format numérique (par exemple « %.3f »)
- **lbl** – si True, affiche les étiquettes de forme
- **option** – shape style option (« s » for signal, « i » for image)

**Renvoie**

Item graphique

```
class cdl.obj.ShapeTypes(value, names=None, *, module=None, qualname=None, type=None, start=1,
                        boundary=None)
```

Types de formes pour les métadonnées d'image

**RECTANGLE** = **'\_rec\_'**

Forme rectangulaire

**CIRCLE** = **'\_cir\_'**

Forme circulaire

**ELLIPSE** = **'\_ell\_'**

Forme elliptique

**SEGMENT** = **'\_seg\_'**

Forme segment

**MARKER** = **'\_mar\_'**

Forme marqueur

**POINT** = **'\_poi\_'**

Forme point

**POLYGON** = **'\_pol\_'**

Forme polygonale

```
class cdl.obj.UniformRandomParam(title=None, comment=None, icon="")
```

Paramètres de signal/image aléatoire de loi uniforme

**apply\_integer\_range**(vmin, vmax)

Fait quelque chose en cas de plage min-max entière

```
class cdl.obj.NormalRandomParam(title=None, comment=None, icon="")
```

Paramètres de signal/image aléatoire de loi normale

**apply\_integer\_range**(vmin, vmax)

Fait quelque chose en cas de plage min-max entière

```
class cdl.obj.BaseProcParam(title=None, comment=None, icon="")
```

Base class for processing parameters

**apply\_integer\_range**(vmin, vmax)

Fait quelque chose en cas de plage min-max entière

**apply\_float\_range**(vmin, vmax)

Do something in case of float min-max range

**set\_from\_datatype**(dtype)

Set min/max range from NumPy datatype

### 3.3.2 Modèle de signal

```
class cdl.obj.SignalObj
    Objet signal

    uuid
        Par défaut : None.
        Type
            guidata.dataset.dataitems.StringItem

    xydata
        Données. Par défaut : None.
        Type
            guidata.dataset.dataitems.FloatArrayItem

    metadata
        Métadonnées. Par défaut : {}.
        Type
            guidata.dataset.dataitems.DictItem

    title
        Titre du signal. Par défaut : "Sans titre".
        Type
            guidata.dataset.dataitems.StringItem

    xlabel
        Titre. Par défaut : "".
        Type
            guidata.dataset.dataitems.StringItem

    xunit
        Unité. Par défaut : "".
        Type
            guidata.dataset.dataitems.StringItem

    ylabel
        Titre. Par défaut : "".
        Type
            guidata.dataset.dataitems.StringItem

    yunit
        Unité. Par défaut : "".
        Type
            guidata.dataset.dataitems.StringItem

    autoscale
        Par défaut : True.
        Type
            guidata.dataset.dataitems.BoolItem

    xscalelog
        Par défaut : False.
        Type
            guidata.dataset.dataitems.BoolItem
```

**xscalemin**

Borne inférieure. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**xscalemax**

Borne supérieure. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**yscalelog**

Par défaut : False.

**Type**

`guidata.dataset.dataitems.BoolItem`

**yscalemin**

Borne inférieure. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**yscalemax**

Borne supérieure. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(uuid : str, xydata : numpy.ndarray, metadata : dict, title : str, xlabel : str, xunit : str, ylabel : str, yunit : str, autoscale : bool, xscalelog : bool, xscalemin : float, xscalemax : float, yscalelog : bool, yscalemin : float, yscalemax : float) → cdl.core.model.signal.SignalObj`

Renvoie une nouvelle instance de `SignalObj` avec les champs initialisés aux valeurs données.

**Paramètres**

- **uuid** (*str*) – Par défaut : None.
- **xydata** (*numpy.ndarray*) – Données. Par défaut : None.
- **metadata** (*dict*) – Métadonnées. Par défaut : {}.
- **title** (*str*) – Titre du signal. Par défaut : “Sans titre”.
- **xlabel** (*str*) – Titre. Par défaut : “”.
- **xunit** (*str*) – Unité. Par défaut : “”.
- **ylabel** (*str*) – Titre. Par défaut : “”.
- **yunit** (*str*) – Unité. Par défaut : “”.
- **autoscale** (*bool*) – Par défaut : True.
- **xscalelog** (*bool*) – Par défaut : False.
- **xscalemin** (*float*) – Borne inférieure. Par défaut : None.
- **xscalemax** (*float*) – Borne supérieure. Par défaut : None.
- **yscalelog** (*bool*) – Par défaut : False.
- **yscalemin** (*float*) – Borne inférieure. Par défaut : None.
- **yscalemax** (*float*) – Borne supérieure. Par défaut : None.

**Renvoie**

Nouvelle instance de `SignalObj`.

**static** `get_roi_class() → Type[SignalROI]`

Return ROI class

**regenerate\_uuid()**

Regénère l’UUID

Cette méthode est utilisée pour régénérer l'UUID après avoir chargé l'objet à partir d'un fichier. Cela est nécessaire pour éviter les conflits d'UUID lors du chargement d'objets à partir d'un fichier sans effacer l'espace de travail au préalable.

**copy**(*title* : *str* | *None* = *None*, *dtype* : *dtype* | *None* = *None*) → *SignalObj*

Copie l'objet.

**Paramètres**

- **title** – titre
- **dtype** – type de données

**Renvoie**

Objet copié

**set\_data\_type**(*dtype* : *dtype*) → *None*

Change le type de données.

**Paramètres**

**type** (*Data*) –

**set\_xydata**(*x* : *ndarray* | *list*, *y* : *ndarray* | *list*, *dx* : *ndarray* | *list* | *None* = *None*, *dy* : *ndarray* | *list* | *None* = *None*) → *None*

Définit les données xy

**Paramètres**

- **x** – données x
- **y** – données y
- **dx** – données dx (optionnel : barres d'erreur)
- **dy** – données dy (optionnel : barres d'erreur)

**property x**: *ndarray* | *None*

Récupère les données x

**property y**: *ndarray* | *None*

Récupère les données y

**property data**: *ndarray* | *None*

Récupère les données y

**property dx**: *ndarray* | *None*

Récupère les données dx

**property dy**: *ndarray* | *None*

Récupère les données dy

**get\_data**(*roi\_index* : *int* | *None* = *None*) → *tuple*[*ndarray*, *ndarray*]

Renvoie les données originales (si la ROI n'est pas définie ou si l'*roi\_index* est *None*), ou les données de la ROI (si la ROI et l'*roi\_index* sont définis).

**Paramètres**

**roi\_index** – index de la ROI

**Renvoie**

Données

**update\_plot\_item\_parameters**(*item* : *CurveItem*) → *None*

Met à jour les paramètres de l'élément de tracé à partir des données/métadonnées de l'objet

Prend en compte un sous-ensemble de paramètres d'élément de tracé. Ces paramètres peuvent avoir été remplacés par des entrées de métadonnées d'objet ou d'autres données d'objet. Le but est de mettre à jour l'élément de tracé en conséquence.

C'est *presque* l'opération inverse de *update\_metadata\_from\_plot\_item*.

**Paramètres****item** – élément de tracé**update\_metadata\_from\_plot\_item**(*item* : *CurveItem*) → *None*

Met à jour les métadonnées à partir de l'élément de tracé.

Prend en compte un sous-ensemble de paramètres d'élément de tracé. Ces paramètres peuvent avoir été modifiés par l'utilisateur via l'interface graphique de l'élément de tracé. Le but est de mettre à jour les métadonnées en conséquence.

C'est *presque* l'opération inverse de *update\_plot\_item\_parameters*.

**Paramètres****item** – élément de tracé**make\_item**(*update\_from* : *CurveItem* | *None* = *None*) → *CurveItem*

Crée un élément de tracé à partir des données.

**Paramètres****update\_from** – élément de tracé à mettre à jour à partir de**Renvoie**

Item graphique

**update\_item**(*item* : *CurveItem*, *data\_changed* : *bool* = *True*) → *None*

Met à jour l'élément de tracé à partir des données.

**Paramètres**— **item** – élément de tracé— **data\_changed** – si *True*, les données ont changé**physical\_to\_indices**(*coords* : *list[float]* | *ndarray*) → *ndarray*

Convert coordinates from physical (real world) to (array) indices (pixel)

**Paramètres****coords** – coordonnées**Renvoie**

Indices

**indices\_to\_physical**(*indices* : *list[int]* | *ndarray*) → *ndarray*

Convert coordinates from (array) indices to physical (real world)

**Paramètres****indices** – indices**Renvoie**

Coordinates

**add\_label\_with\_title**(*title* : *str* | *None* = *None*) → *None*

Ajouter une étiquette avec une annotation de titre

**Paramètres****title** – titre (si *None*, utiliser le titre du signal)**accept**(*vis* : *object*) → *None*

Fonction d'aide qui transmet le visiteur aux méthodes accept des éléments dans cet ensemble de données

**Paramètres****vis** (*object*) – objet visiteur**add\_annotations\_from\_file**(*filename* : *str*) → *None*

Ajouter des annotations d'objet à partir d'un fichier (JSON).

**Paramètres****filename** – nom du fichier

**add\_annotations\_from\_items**(*items* : *list*) → *None*

Ajouter des annotations d'objet (éléments de tracé d'annotation).

**Paramètres**

**items** – éléments de tracé d'annotation

**property annotations**: *str*

Obtenir les annotations d'objet (chaîne JSON décrivant les éléments de tracé d'annotation)

**check**() → *list[str]*

Vérifier les valeurs des éléments de l'ensemble de données

**Renvoie**

liste des erreurs

**Type renvoyé**

*list[str]*

**check\_data**()

Vérifier si les données sont valides, lever une exception si ce n'est pas le cas

**Lève**

**TypeError** – si le type de données n'est pas pris en charge

**delete\_results**() → *None*

Delete all object results (shapes and properties)

**deserialize**(*reader* : *HDF5Reader* | *JSONReader* | *INIRReader*) → *None*

Désérialiser l'ensemble de données

**Paramètres**

**reader** (*HDF5Reader* | *JSONReader* | *INIRReader*) – objet lecteur

**edit**(*parent* : *QWidget* | *None* = *None*, *apply* : *Callable* | *None* = *None*, *wordwrap* : *bool* = *True*, *size* : *QSize* | *tuple[int, int]* | *None* = *None*) → *DataSetEditDialog*

Ouvrir une boîte de dialogue pour modifier l'ensemble de données

**Paramètres**

— **parent** – widget parent (par défaut, *None* signifie pas de parent)

— **apply** – callback d'application (par défaut, *None*)

— **wordwrap** – si *True*, le texte du commentaire est enveloppé

— **size** – taille de la boîte de dialogue (objet *QSize* ou tuple d'entiers (largeur, hauteur))

**get\_comment**() → *str* | *None*

Renvoyer le commentaire de l'ensemble de données

**Renvoie**

commentaire

**Type renvoyé**

*str* | *None*

**get\_icon**() → *str* | *None*

Renvoyer l'icône de l'ensemble de données

**Renvoie**

icône

**Type renvoyé**

*str* | *None*

**get\_items**(*copy*=*False*) → *list[DataItem]*

Renvoie tous les objets *DataItem* de l'instance *DataSet*. Ignore les objets privés qui ont un nom commençant par un tiret bas (par exemple “\_private\_item = ...”)

**Paramètres**

- **copy** – Si True, effectue une copie profonde de la liste DataItem, sinon renvoie l’original.
- **False.** (*Defaults to*) –

**Renvoie**

`_description_`

**get\_masked\_view()** → MaskedArray

Retourne une vue masquée des données

**Renvoie**

Vue masquée

**get\_metadata\_option(name : str)** → Any

Renvoyer la valeur de l’option de métadonnées

Une option de métadonnées est une entrée de métadonnées commençant par un underscore. C’est un moyen de stocker des options spécifiques à l’application dans les métadonnées de l’objet.

**Paramètres**

**name** – nom de l’option

**Renvoie**

Valeur de l’option

**Noms d’options valides :**

“format” : chaîne de format “showlabel” : afficher l’étiquette

**get\_title()** → str

Renvoyer le titre de l’ensemble de données

**Renvoie**

titre

**Type renvoyé**

str

**classmethod get\_valid\_dtypenames()** → list[str]

Obtenir les noms de types de données valides

**Renvoie**

Noms de types de données valides pris en charge par cette classe

**invalidate\_maskdata\_cache()** → None

Invalider le cache des données masquées : force la reconstruction

**iterate\_resultproperties()** → Iterable[ResultProperties]

Iterate over object result properties.

**Produit (yield)**

Result properties

**iterate\_resultshapes()** → Iterable[ResultShape]

Itérer sur les formes de résultat de l’objet.

**Produit (yield)**

Forme résultat

**iterate\_roi\_indices()** → Generator[int | None, None, None]

Iterate over object ROI indices (if there is no ROI, yield None)

**iterate\_shape\_items(editable : bool = False)**

Iterate over shape items encoded in metadata (if any).

**Paramètres**

**editable** – if True, annotations are editable

**Produit (yield)**

Item graphique

**property maskdata:** `ndarray`

Renvoie les données masquées (zones en dehors des régions d'intérêt définies)

**Renvoie**

Données masquées

**property number:** `int`

Renvoyer le numéro de l'objet (utilisé pour l'ID court)

**read\_config**(*conf* : `UserConfig`, *section* : `str`, *option* : `str`) → `None`

Lire la configuration à partir d'une instance UserConfig

**Paramètres**

— **conf** (`UserConfig`) – instance UserConfig

— **section** (`str`) – nom de la section

— **option** (`str`) – nom de l'option

**remove\_all\_shapes**() → `None`

Supprimer les formes de métadonnées et les ROI

**reset\_metadata\_to\_defaults**() → `None`

Réinitialiser les métadonnées aux valeurs par défaut

**restore\_attr\_from\_metadata**(*attrname* : `str`, *default* : `Any`) → `None`

Restore attribute from metadata

**Paramètres**

— **attrname** – attribute name

— **default** – default value

**property roi:** `TypeROI` | `None`

Return object regions of interest object.

**Renvoie**

Regions of interest object

**roi\_has\_changed**() → `bool`

Renvoie True si la ROI a changé depuis le dernier appel à cette méthode.

Le premier appel à cette méthode renverra True si la ROI n'a pas encore été définie, ou si la ROI a été définie et a changé depuis le dernier appel à cette méthode. Le prochain appel à cette méthode renverra toujours False si la ROI n'a pas changé entre-temps.

**Renvoie**

True si la ROI a changé

**save\_attr\_to\_metadata**(*attrname* : `str`, *new\_value* : `Any`) → `None`

Save attribute to metadata

**Paramètres**

— **attrname** – attribute name

— **new\_value** – new value

**serialize**(*writer* : `HDF5Writer` | `JSONWriter` | `INIWriter`) → `None`

Sérialiser le jeu de données

**Paramètres**

**writer** (`HDF5Writer` | `JSONWriter` | `INIWriter`) – objet écrivain

**set\_defaults()** → *None*

Définir les valeurs par défaut

**classmethod set\_global\_prop**(*realm : str, \*\*kwargs*) → *None*

Définir les propriétés globales pour tous les éléments de données du jeu de données

**Paramètres**

- **realm** (*str*) – nom du domaine
- **kwargs** (*dict*) – propriétés à définir

**set\_metadata\_option**(*name : str, value : Any*) → *None*

Définir la valeur de l’option de métadonnées

Une option de métadonnées est une entrée de métadonnées commençant par un underscore. C’est un moyen de stocker des options spécifiques à l’application dans les métadonnées de l’objet.

**Paramètres**

- **name** – nom de l’option
- **value** – valeur de l’option

**Noms d’options valides :**

“format” : chaîne de format “showlabel” : afficher l’étiquette

**property short\_id:** *str*

ID court de l’objet

**text\_edit()** → *None*

Modifier le jeu de données avec une saisie de texte uniquement

**to\_string**(*debug : bool | None = False, indent : str | None = None, align : bool | None = False, show\_hidden : bool | None = True*) → *str*

Renvoie une représentation lisible du jeu de données. Si debug est True, ajoute plus de détails sur les éléments de données

**Paramètres**

- **debug** (*bool*) – si True, ajoute plus de détails sur les éléments de données
- **indent** (*str*) – chaîne d’indentation (par défaut, None signifie aucune indentation)
- **align** (*bool*) – si True, aligne les éléments de données (par défaut, False)
- **show\_hidden** (*bool*) – si True, affiche les éléments de données masqués (par défaut, True)

**Renvoie**

représentation sous forme de chaîne du jeu de données

**Type renvoyé**

*str*

**transform\_shapes**(*orig, func, param=None*)

Appliquer la fonction de transformation aux coordonnées de la forme / des annotations du résultat.

**Paramètres**

- **orig** – objet d’origine
- **func** – fonction de transformation
- **param** – paramètre de la fonction de transformation

**update\_metadata\_from**(*other\_metadata : dict[str, Any]*) → *None*

Update metadata from another object’s metadata (merge result shapes and annotations, and update the rest of the metadata).

**Paramètres**

**other\_metadata** – other object metadata

**update\_metadata\_view\_settings()** → *None*

Mettre à jour les paramètres d’affichage des métadonnées à partir de `Conf.view`

**update\_resultshapes\_from**(*other* : *TypeObj*) → *None*

Mettre à jour la forme géométrique à partir d’un autre objet (fusionner les métadonnées).

**Paramètres**

**other** – autre objet, à partir duquel mettre à jour cet objet

**view**(*parent* : *QWidget* | *None* = *None*, *wordwrap* : *bool* = *True*, *size* : *QSize* | *tuple*[*int*, *int*] | *None* = *None*) → *None*

Ouvrir une boîte de dialogue pour afficher le jeu de données

**Paramètres**

— **parent** – widget parent (par défaut, *None* signifie pas de parent)

— **wordwrap** – si *True*, le texte du commentaire est enveloppé

— **size** – taille de la boîte de dialogue (objet *QSize* ou tuple d’entiers (largeur, hauteur))

**write\_config**(*conf* : *UserConfig*, *section* : *str*, *option* : *str*) → *None*

Écrire la configuration dans une instance *UserConfig*

**Paramètres**

— **conf** (*UserConfig*) – instance *UserConfig*

— **section** (*str*) – nom de la section

— **option** (*str*) – nom de l’option

`cdl.obj.read_signal`(*filename* : *str*) → *SignalObj*

Lire un signal à partir d’un fichier.

**Paramètres**

**filename** – Nom de fichier.

**Renvoie**

Signal.

`cdl.obj.read_signals`(*filename* : *str*) → *list*[*SignalObj*]

Lire une liste de signaux à partir d’un fichier.

**Paramètres**

**filename** – Nom de fichier.

**Renvoie**

Liste de signaux.

`cdl.obj.create_signal_roi`(*coords* : *ndarray* | *list*[*float*, *float*] | *list*[*list*[*float*, *float*]], *indices* : *bool* = *False*, *singleobj* : *bool* | *None* = *None*, *inverse* : *bool* = *False*, *title* : *str* = "") → *SignalROI*

Create Signal Regions of Interest (ROI) object. More ROIs can be added to the object after creation, using the *add\_roi* method.

**Paramètres**

— **coords** – single ROI coordinates [*xmin*, *xmax*], or multiple ROIs coordinates [[*xmin1*, *xmax1*], [*xmin2*, *xmax2*], ...] (lists or NumPy arrays)

— **indices** – if *True*, coordinates are indices, if *False*, they are physical values (default to *False* for signals)

— **singleobj** – if *True*, when extracting data defined by ROIs, only one object is created (default to *True*). If *False*, one object is created per single ROI. If *None*, the value is get from the user configuration

— **inverse** – if *True*, ROI is outside the region

— **title** – titre

**Renvoie**

Regions of Interest (ROI) object

**Lève****ValueError** – if the number of coordinates is not even

```
cdl.obj.create_signal(title : str, x : ndarray | None = None, y : ndarray | None = None, dx : ndarray | None =
None, dy : ndarray | None = None, metadata : dict | None = None, units : tuple[str, str] |
None = None, labels : tuple[str, str] | None = None) → SignalObj
```

Créer un nouvel objet Signal.

**Paramètres**

- **title** – titre du signal
- **x** – données X
- **y** – données Y
- **dx** – données dX (facultatif : barres d'erreur)
- **dy** – données dY (facultatif : barres d'erreur)
- **metadata** – métadonnées du signal
- **units** – unités X, Y (tuple de chaînes)
- **labels** – étiquettes X, Y (tuple de chaînes)

**Renvoie**

Objet signal

```
cdl.obj.create_signal_from_param(newparam : NewSignalParam, addparam : DataSet | None = None, edit :
bool = False, parent : QWidget | None = None) → SignalObj | None
```

Créer un nouvel objet Signal à partir d'une boîte de dialogue.

**Paramètres**

- **newparam** – nouveaux paramètres du signal
- **addparam** – paramètres supplémentaires
- **edit** – Ouvrir une boîte de dialogue pour modifier les paramètres (par défaut : False)
- **parent** – widget parent

**Renvoie**

Objet Signal ou None si annulé

```
cdl.obj.new_signal_param(title : str | None = None, stype : str | None = None, xmin : float | None = None,
xmax : float | None = None, size : int | None = None) → NewSignalParam
```

Créer une nouvelle instance de jeu de données Signal.

**Paramètres**

- **title** – titre du jeu de données (par défaut : None, utilise le titre par défaut)
- **stype** – type de signal (par défaut : None, utilise le type par défaut)
- **xmin** – X min (par défaut : None, utilise la valeur par défaut)
- **xmax** – X max (par défaut : None, utilise la valeur par défaut)
- **size** – taille du signal (par défaut : None, utilise la valeur par défaut)

**Renvoie**

nouvelle instance de jeu de données signal

**Type renvoyé***NewSignalParam*

```
class cdl.obj.SignalTypes(value, names=None, *, module=None, qualname=None, type=None, start=1,
boundary=None)
```

Types de signal

**ZEROS** = 'zéros'

Signal rempli de zéros

**GAUSS** = 'gaussienne'

Fonction gaussienne

**LORENTZ** = 'lorentzienne'

Fonction lorentzienne

**VOIGT** = 'Voigt'

Fonction de Voigt

**UNIFORMRANDOM** = 'aléatoire (loi uniforme)'

Signal aléatoire (loi uniforme)

**NORMALRANDOM** = 'aléatoire (loi normale)'

Signal aléatoire (loi normale)

**SINUS** = 'sinus'

Fonction sinus

**COSINUS** = 'cosinus'

Fonction cosinus

**SAWTOOTH** = 'dent de scie'

Fonction en dents de scie

**TRIANGLE** = 'triangle'

Fonction triangle

**SQUARE** = 'carré'

Fonction carrée

**SINC** = 'sinus cardinal'

Sinus cardinal

**STEP** = 'échelon'

Fonction échelon

**EXPONENTIAL** = 'exponentielle'

Exponential function

**PULSE** = 'impulsion'

Pulse function

**POLYNOMIAL** = 'polynomial'

Polynomial function

**EXPERIMENTAL** = 'expérimental'

Experimental function

**class** cdl.obj.NewSignalParam

Nouveau jeu de données signal

**title**

Titre. Par défaut : None.

**Type**

`guidata.dataset.dataitems.StringItem`

**xmin**

Par défaut : -10.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**xmax**

Par défaut : 10.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**size**

Taille. Nombre de points. Entier supérieur à 1. Par défaut : 500.

**Type**

`guidata.dataset.dataitems.IntItem`

**stype**

Sélection unique parmi : `SignalTypes.ZEROS`, `SignalTypes.GAUSS`, `SignalTypes.LORENTZ`, `SignalTypes.VOIGT`, `SignalTypes.UNIFORMRANDOM`, `SignalTypes.NORMALRANDOM`, `SignalTypes.SINUS`, `SignalTypes.COSINUS`, `SignalTypes.SAWTOOTH`, `SignalTypes.TRIANGLE`, `SignalTypes.SQUARE`, `SignalTypes.SINC`, `SignalTypes.STEP`, `SignalTypes.EXPONENTIAL`, `SignalTypes.PULSE`, `SignalTypes.POLYNOMIAL`, `SignalTypes.EXPERIMENTAL`. Par défaut : `SignalTypes.ZEROS`.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**classmethod** `create(title : str, xmin : float, xmax : float, size : int, stype :`  
`cdl.core.model.signal.SignalTypes) → cdl.core.model.signal.NewSignalParam`

Renvoie une nouvelle instance de `NewSignalParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **title** (*str*) – Titre. Par défaut : None.
- **xmin** (*float*) – Par défaut : -10.0.
- **xmax** (*float*) – Par défaut : 10.0.
- **size** (*int*) – Taille. Nombre de points. Entier supérieur à 1. Par défaut : 500.
- **stype** (`cdl.core.model.signal.SignalTypes`) – Sélection unique parmi : `SignalTypes.ZEROS`, `SignalTypes.GAUSS`, `SignalTypes.LORENTZ`, `SignalTypes.VOIGT`, `SignalTypes.UNIFORMRANDOM`, `SignalTypes.NORMALRANDOM`, `SignalTypes.SINUS`, `SignalTypes.COSINUS`, `SignalTypes.SAWTOOTH`, `SignalTypes.TRIANGLE`, `SignalTypes.SQUARE`, `SignalTypes.SINC`, `SignalTypes.STEP`, `SignalTypes.EXPONENTIAL`, `SignalTypes.PULSE`, `SignalTypes.POLYNOMIAL`, `SignalTypes.EXPERIMENTAL`. Par défaut : `SignalTypes.ZEROS`.

**Renvoie**

Nouvelle instance de `NewSignalParam`.

**class** `cdl.obj.GaussLorentzVoigtParam`

Paramètres pour les fonctions gaussiennes et lorentziennes

**a**

Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**ymin**

Par défaut : 0.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**sigma**

. Par défaut : 1.0.

**Type**`guidata.dataset.dataitems.FloatItem`**mu**

. Par défaut : 0.0.

**Type**`guidata.dataset.dataitems.FloatItem`

**classmethod create**(*a* : *float*, *ymin* : *float*, *sigma* : *float*, *mu* : *float*) → *cdl.core.model.signal.GaussLorentzVoigtParam*

Renvoie une nouvelle instance de *GaussLorentzVoigtParam* avec les champs initialisés aux valeurs données.

**Paramètres**

- **a** (*float*) – Par défaut : 1.0.
- **ymin** (*float*) – Par défaut : 0.0.
- **sigma** (*float*) – . Par défaut : 1.0.
- **mu** (*float*) – . Par défaut : 0.0.

**Renvoie**Nouvelle instance de *GaussLorentzVoigtParam*.

**class** `cdl.obj.StepParam`

Paramètres pour la fonction d'étape

**a1**

Par défaut : 0.0.

**Type**`guidata.dataset.dataitems.FloatItem`**a2**

Par défaut : 1.0.

**Type**`guidata.dataset.dataitems.FloatItem`**x0**

Par défaut : 0.0.

**Type**`guidata.dataset.dataitems.FloatItem`

**classmethod create**(*a1* : *float*, *a2* : *float*, *x0* : *float*) → *cdl.core.model.signal.StepParam*

Renvoie une nouvelle instance de *StepParam* avec les champs initialisés aux valeurs données.

**Paramètres**

- **a1** (*float*) – Par défaut : 0.0.
- **a2** (*float*) – Par défaut : 1.0.
- **x0** (*float*) – Par défaut : 0.0.

**Renvoie**Nouvelle instance de *StepParam*.

**class** `cdl.obj.PeriodicParam`

Paramètres pour les fonctions périodiques

**a**

Par défaut : 1.0.

**Type**`guidata.dataset.dataitems.FloatItem`

**ymin**

Par défaut : 0.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**freq**

Fréquence. Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**freq\_unit**

Unité. Sélection unique parmi : `FreqUnits.HZ`, `FreqUnits.KHZ`, `FreqUnits.MHZ`, `FreqUnits.GHZ`. Par défaut : `FreqUnits.HZ`.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**phase**

Flottant, unité : °. Par défaut : 0.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(a : float, ymin : float, freq : float, freq_unit : cdl.core.model.signal.FreqUnits, phase : float) → cdl.core.model.signal.PeriodicParam`

Renvoie une nouvelle instance de *PeriodicParam* avec les champs initialisés aux valeurs données.

**Paramètres**

- **a** (*float*) – Par défaut : 1.0.
- **ymin** (*float*) – Par défaut : 0.0.
- **freq** (*float*) – Fréquence. Par défaut : 1.0.
- **freq\_unit** (*cdl.core.model.signal.FreqUnits*) – Unité. Sélection unique parmi : `FreqUnits.HZ`, `FreqUnits.KHZ`, `FreqUnits.MHZ`, `FreqUnits.GHZ`. Par défaut : `FreqUnits.HZ`.
- **phase** (*float*) – Flottant, unité : °. Par défaut : 0.0.

**Renvoie**

Nouvelle instance de *PeriodicParam*.

**get\_frequency\_in\_hz()**

Renvoie la fréquence en Hz

**class** `cdl.obj.ROI1DParam`

Signal ROI parameters

**xmin**

Coordonnée du premier point. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**xmax**

Coordonnée du dernier point. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(xmin : float, xmax : float) → cdl.core.model.signal.ROI1DParam`

Renvoie une nouvelle instance de *ROI1DParam* avec les champs initialisés aux valeurs données.

**Paramètres**

- **xmin** (*float*) – Coordonnée du premier point. Par défaut : None.
- **xmax** (*float*) – Coordonnée du dernier point. Par défaut : None.

**Renvoie**

Nouvelle instance de *ROI1DParam*.

**to\_single\_roi**(*obj* : *SignalObj*, *title* : *str* = "") → *SegmentROI*

Convert parameters to single ROI

**Paramètres**

- **obj** – signal object
- **title** – ROI title

**Renvoie**

Single ROI

**get\_data**(*obj* : *SignalObj*) → *ndarray*

Get signal data in ROI

**Paramètres**

- obj** – signal object

**Renvoie**

Data in ROI

**class** *cdl.obj.SignalROI*(*singleobj* : *bool* | *None* = *None*, *inverse* : *bool* = *False*)

Signal Regions of Interest

**Paramètres**

- **singleobj** – if True, when extracting data defined by ROIs, only one object is created (default to True). If False, one object is created per single ROI. If None, the value is get from the user configuration
- **inverse** – if True, ROI is outside the region

**static get\_compatible\_single\_roi\_classes**() → *list*[*Type*[*SegmentROI*]]

Return compatible single ROI classes

**to\_mask**(*obj* : *SignalObj*) → *ndarray*[*bool*]

Create mask from ROI

**Paramètres**

- obj** – signal object

**Renvoie**

Mask (boolean array where True values are inside the ROI)

### 3.3.3 Modèle d'image

**class** *cdl.obj.ImageObj*

Objet image

**uuid**

Par défaut : None.

**Type**

*guidata.dataset.dataitems.StringItem*

**data**

Données. Par défaut : None.

**Type**

*guidata.dataset.dataitems.FloatArrayItem*

**metadata**

Métadonnées. Par défaut : {}.

**Type**

`guidata.dataset.dataitems.DictItem`

**x0**

$X_0$ . Par défaut : 0.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**y0**

$Y_0$ . Par défaut : 0.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**dx**

x. Flottant, non nul. Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**dy**

y. Flottant, non nul. Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**title**

Titre de l'image. Par défaut : "Sans titre".

**Type**

`guidata.dataset.dataitems.StringItem`

**xlabel**

Titre. Par défaut : "".

**Type**

`guidata.dataset.dataitems.StringItem`

**xunit**

Unité. Par défaut : "".

**Type**

`guidata.dataset.dataitems.StringItem`

**ylabel**

Titre. Par défaut : "".

**Type**

`guidata.dataset.dataitems.StringItem`

**yunit**

Unité. Par défaut : "".

**Type**

`guidata.dataset.dataitems.StringItem`

**zlabel**

Titre. Par défaut : "".

**Type**

`guidata.dataset.dataitems.StringItem`

**zunit**

Unité. Par défaut : "".

**Type**

`guidata.dataset.dataitems.StringItem`

**autoscale**

Par défaut : True.

**Type**

`guidata.dataset.dataitems.BoolItem`

**xscalelog**

Par défaut : False.

**Type**

`guidata.dataset.dataitems.BoolItem`

**xscalemin**

Borne inférieure. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**xscalemax**

Borne supérieure. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**yscalelog**

Par défaut : False.

**Type**

`guidata.dataset.dataitems.BoolItem`

**yscalemin**

Borne inférieure. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**yscalemax**

Borne supérieure. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**zscalemin**

Borne inférieure. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**zscalemax**

Borne supérieure. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod create**(*uuid* : *str*, *data* : *numpy.ndarray*, *metadata* : *dict*, *x0* : *float*, *y0* : *float*, *dx* : *float*, *dy* : *float*, *title* : *str*, *xlabel* : *str*, *xunit* : *str*, *ylabel* : *str*, *yunit* : *str*, *zlabel* : *str*, *zunit* : *str*, *autoscale* : *bool*, *xscalelog* : *bool*, *xscalemin* : *float*, *xscalemax* : *float*, *yscalelog* : *bool*, *yscalemin* : *float*, *yscalemax* : *float*, *zscalemin* : *float*, *zscalemax* : *float*) → *cdl.core.model.image.ImageObj*

Renvoie une nouvelle instance de *ImageObj* avec les champs initialisés aux valeurs données.

#### Paramètres

- **uuid** (*str*) – Par défaut : None.
- **data** (*numpy.ndarray*) – Données. Par défaut : None.
- **metadata** (*dict*) – Métadonnées. Par défaut : {}.
- **x0** (*float*) –  $X_0$ . Par défaut : 0.0.
- **y0** (*float*) –  $Y_0$ . Par défaut : 0.0.
- **dx** (*float*) –  $x$ . Flottant, non nul. Par défaut : 1.0.
- **dy** (*float*) –  $y$ . Flottant, non nul. Par défaut : 1.0.
- **title** (*str*) – Titre de l’image. Par défaut : “Sans titre”.
- **xlabel** (*str*) – Titre. Par défaut : “”.
- **xunit** (*str*) – Unité. Par défaut : “”.
- **ylabel** (*str*) – Titre. Par défaut : “”.
- **yunit** (*str*) – Unité. Par défaut : “”.
- **zlabel** (*str*) – Titre. Par défaut : “”.
- **zunit** (*str*) – Unité. Par défaut : “”.
- **autoscale** (*bool*) – Par défaut : True.
- **xscalelog** (*bool*) – Par défaut : False.
- **xscalemin** (*float*) – Borne inférieure. Par défaut : None.
- **xscalemax** (*float*) – Borne supérieure. Par défaut : None.
- **yscalelog** (*bool*) – Par défaut : False.
- **yscalemin** (*float*) – Borne inférieure. Par défaut : None.
- **yscalemax** (*float*) – Borne supérieure. Par défaut : None.
- **zscalemin** (*float*) – Borne inférieure. Par défaut : None.
- **zscalemax** (*float*) – Borne supérieure. Par défaut : None.

#### Renvoie

Nouvelle instance de *ImageObj*.

**static get\_roi\_class()** → *Type[ImageROI]*

Return ROI class

**regenerate\_uuid()**

Regénère l’UUID

Cette méthode est utilisée pour régénérer l’UUID après avoir chargé l’objet à partir d’un fichier. Cela est nécessaire pour éviter les conflits d’UUID lors du chargement d’objets à partir d’un fichier sans effacer l’espace de travail au préalable.

**set\_metadata\_from**(*obj* : *Mapping* | *dict*) → None

Définir les métadonnées à partir de l’objet : semblable à un dictionnaire

#### Paramètres

**obj** – objet

**property dicom\_template**

Obtenir le modèle DICOM

**property width:** *float*

Return image width, i.e. number of columns multiplied by pixel size

**property height:** *float*

Return image height, i.e. number of rows multiplied by pixel size

**property xc:** *float*

Renvoie la coordonnée X du centre de l’image

**property** `yc`: `float`

Renvoie la coordonnée Y du centre de l'image

**get\_data**(`roi_index` : `int` | `None` = `None`) → `ndarray`

Renvoie les données originales (si la ROI n'est pas définie ou si l'`roi_index` est `None`), ou les données de la ROI (si la ROI et l'`roi_index` sont définis).

**Paramètres**

**roi\_index** – index de la ROI

**Renvoie**

Données masquées

**copy**(`title` : `str` | `None` = `None`, `dtype` : `dtype` | `None` = `None`) → `ImageObj`

Copie l'objet.

**Paramètres**

— **title** – titre

— **dtype** – type de données

**Renvoie**

Objet copié

**set\_data\_type**(`dtype` : `dtype`) → `None`

Change data type. If data type is integer, clip values to the new data type's range, thus avoiding overflow or underflow.

**Paramètres**

**type** (`Data`) –

**update\_plot\_item\_parameters**(`item` : `MaskedImageItem`) → `None`

Met à jour les paramètres de l'élément de tracé à partir des données/métadonnées de l'objet

Prend en compte un sous-ensemble de paramètres d'élément de tracé. Ces paramètres peuvent avoir été remplacés par des entrées de métadonnées d'objet ou d'autres données d'objet. Le but est de mettre à jour l'élément de tracé en conséquence.

C'est *presque* l'opération inverse de `update_metadata_from_plot_item`.

**Paramètres**

**item** – élément de tracé

**update\_metadata\_from\_plot\_item**(`item` : `MaskedImageItem`) → `None`

Met à jour les métadonnées à partir de l'élément de tracé.

Prend en compte un sous-ensemble de paramètres d'élément de tracé. Ces paramètres peuvent avoir été modifiés par l'utilisateur via l'interface graphique de l'élément de tracé. Le but est de mettre à jour les métadonnées en conséquence.

C'est *presque* l'opération inverse de `update_plot_item_parameters`.

**Paramètres**

**item** – élément de tracé

**make\_item**(`update_from` : `MaskedImageItem` | `None` = `None`) → `MaskedImageItem`

Crée un élément de tracé à partir des données.

**Paramètres**

**update\_from** – mettre à jour à partir de l'élément de tracé

**Renvoie**

Item graphique

**update\_item**(`item` : `MaskedImageItem`, `data_changed` : `bool` = `True`) → `None`

Met à jour l'élément de tracé à partir des données.

**Paramètres**

- **item** – élément de tracé
- **data\_changed** – si True, les données ont changé

**physical\_to\_indices**(*coords* : *list[float]*) → *ndarray*

Convert coordinates from physical (real world) to (array) indices (pixel)

**Paramètres**

- coords** – coordonnées

**Renvoie**

Indices

**indices\_to\_physical**(*indices* : *list[float | int] | ndarray*) → *ndarray*

Convert coordinates from (array) indices to physical (real world)

**Paramètres**

- indices** – indices

**Renvoie**

Coordinates

**add\_label\_with\_title**(*title* : *str* | *None* = *None*) → *None*

Ajouter une étiquette avec une annotation de titre

**Paramètres**

- title** – titre (si None, utilise le titre de l'image)

**accept**(*vis* : *object*) → *None*

Fonction d'aide qui transmet le visiteur aux méthodes accept des éléments dans cet ensemble de données

**Paramètres**

- vis** (*object*) – objet visiteur

**add\_annotations\_from\_file**(*filename* : *str*) → *None*

Ajouter des annotations d'objet à partir d'un fichier (JSON).

**Paramètres**

- filename** – nom du fichier

**add\_annotations\_from\_items**(*items* : *list*) → *None*

Ajouter des annotations d'objet (éléments de tracé d'annotation).

**Paramètres**

- items** – éléments de tracé d'annotation

**property annotations:** *str*

Obtenir les annotations d'objet (chaîne JSON décrivant les éléments de tracé d'annotation)

**check**() → *list[str]*

Vérifier les valeurs des éléments de l'ensemble de données

**Renvoie**

liste des erreurs

**Type renvoyé**

*list[str]*

**check\_data**()

Vérifier si les données sont valides, lever une exception si ce n'est pas le cas

**Lève**

- TypeError** – si le type de données n'est pas pris en charge

**delete\_results()** → *None*

Delete all object results (shapes and properties)

**deserialize**(*reader : HDF5Reader | JSONReader | INIReader*) → *None*

Désérialiser l'ensemble de données

**Paramètres**

**reader** (*HDF5Reader | JSONReader | INIReader*) – objet lecteur

**edit**(*parent : QWidget | None = None, apply : Callable | None = None, wordwrap : bool = True, size : QSize | tuple[int, int] | None = None*) → *DataSetEditDialog*

Ouvrir une boîte de dialogue pour modifier l'ensemble de données

**Paramètres**

— **parent** – widget parent (par défaut, *None* signifie pas de parent)

— **apply** – callback d'application (par défaut, *None*)

— **wordwrap** – si *True*, le texte du commentaire est enveloppé

— **size** – taille de la boîte de dialogue (objet *QSize* ou tuple d'entiers (largeur, hauteur))

**get\_comment()** → *str | None*

Renvoyer le commentaire de l'ensemble de données

**Renvoie**

commentaire

**Type renvoyé**

*str | None*

**get\_icon()** → *str | None*

Renvoyer l'icône de l'ensemble de données

**Renvoie**

icône

**Type renvoyé**

*str | None*

**get\_items**(*copy=False*) → *list[DataItem]*

Renvoie tous les objets *DataItem* de l'instance *DataSet*. Ignore les objets privés qui ont un nom commençant par un tiret bas (par exemple “\_private\_item = ...”)

**Paramètres**

— **copy** – Si *True*, effectue une copie profonde de la liste *DataItem*, sinon renvoie l'original.

— **False.** (*Defaults to*) –

**Renvoie**

\_description\_

**get\_masked\_view()** → *MaskedArray*

Retourne une vue masquée des données

**Renvoie**

Vue masquée

**get\_metadata\_option**(*name : str*) → *Any*

Renvoyer la valeur de l'option de métadonnées

Une option de métadonnées est une entrée de métadonnées commençant par un underscore. C'est un moyen de stocker des options spécifiques à l'application dans les métadonnées de l'objet.

**Paramètres**

**name** – nom de l'option

**Renvoie**

Valeur de l'option

**Noms d'options valides :**

“format” : chaîne de format “showlabel” : afficher l'étiquette

**get\_title()** → `str`

Renvoyer le titre de l'ensemble de données

**Renvoie**

titre

**Type renvoyé**

`str`

**classmethod get\_valid\_dtypenames()** → `list[str]`

Obtenir les noms de types de données valides

**Renvoie**

Noms de types de données valides pris en charge par cette classe

**invalidate\_maskdata\_cache()** → `None`

Invalider le cache des données masquées : force la reconstruction

**iterate\_resultproperties()** → `Iterable[ResultProperties]`

Iterate over object result properties.

**Produit (yield)**

Result properties

**iterate\_resultshapes()** → `Iterable[ResultShape]`

Itérer sur les formes de résultat de l'objet.

**Produit (yield)**

Forme résultat

**iterate\_roi\_indices()** → `Generator[int | None, None, None]`

Iterate over object ROI indices (if there is no ROI, yield None)

**iterate\_shape\_items(editable : bool = False)**

Iterate over shape items encoded in metadata (if any).

**Paramètres**

**editable** – if True, annotations are editable

**Produit (yield)**

Item graphique

**property maskdata:** `ndarray`

Renvoie les données masquées (zones en dehors des régions d'intérêt définies)

**Renvoie**

Données masquées

**property number:** `int`

Renvoyer le numéro de l'objet (utilisé pour l'ID court)

**read\_config(conf : UserConfig, section : str, option : str)** → `None`

Lire la configuration à partir d'une instance UserConfig

**Paramètres**

— **conf** (`UserConfig`) – instance UserConfig

— **section** (`str`) – nom de la section

— **option** (`str`) – nom de l'option

**remove\_all\_shapes()** → *None*

Supprimer les formes de métadonnées et les ROI

**reset\_metadata\_to\_defaults()** → *None*

Réinitialiser les métadonnées aux valeurs par défaut

**restore\_attr\_from\_metadata**(*attrname* : *str*, *default* : *Any*) → *None*

Restore attribute from metadata

**Paramètres**

- **attrname** – attribute name
- **default** – default value

**property roi**: *TypeROI* | *None*

Return object regions of interest object.

**Renvoie**

Regions of interest object

**roi\_has\_changed()** → *bool*

Renvoie True si la ROI a changé depuis le dernier appel à cette méthode.

Le premier appel à cette méthode renverra True si la ROI n'a pas encore été définie, ou si la ROI a été définie et a changé depuis le dernier appel à cette méthode. Le prochain appel à cette méthode renverra toujours False si la ROI n'a pas changé entre-temps.

**Renvoie**

True si la ROI a changé

**save\_attr\_to\_metadata**(*attrname* : *str*, *new\_value* : *Any*) → *None*

Save attribute to metadata

**Paramètres**

- **attrname** – attribute name
- **new\_value** – new value

**serialize**(*writer* : *HDF5Writer* | *JSONWriter* | *INIWriter*) → *None*

Sérialiser le jeu de données

**Paramètres**

**writer** (*HDF5Writer* | *JSONWriter* | *INIWriter*) – objet écrivain

**set\_defaults()** → *None*

Définir les valeurs par défaut

**classmethod set\_global\_prop**(*realm* : *str*, *\*\*kwargs*) → *None*

Définir les propriétés globales pour tous les éléments de données du jeu de données

**Paramètres**

- **realm** (*str*) – nom du domaine
- **kwargs** (*dict*) – propriétés à définir

**set\_metadata\_option**(*name* : *str*, *value* : *Any*) → *None*

Définir la valeur de l'option de métadonnées

Une option de métadonnées est une entrée de métadonnées commençant par un underscore. C'est un moyen de stocker des options spécifiques à l'application dans les métadonnées de l'objet.

**Paramètres**

- **name** – nom de l'option
- **value** – valeur de l'option

**Noms d'options valides :**

“format” : chaîne de format “showlabel” : afficher l'étiquette

**property short\_id:** `str`

ID court de l'objet

**text\_edit()** → `None`

Modifier le jeu de données avec une saisie de texte uniquement

**to\_string**(*debug* : `bool` | `None` = `False`, *indent* : `str` | `None` = `None`, *align* : `bool` | `None` = `False`,  
*show\_hidden* : `bool` | `None` = `True`) → `str`

Renvoie une représentation lisible du jeu de données. Si `debug` est `True`, ajoute plus de détails sur les éléments de données

**Paramètres**

- **debug** (`bool`) – si `True`, ajoute plus de détails sur les éléments de données
- **indent** (`str`) – chaîne d'indentation (par défaut, `None` signifie aucune indentation)
- **align** (`bool`) – si `True`, aligne les éléments de données (par défaut, `False`)
- **show\_hidden** (`bool`) – si `True`, affiche les éléments de données masqués (par défaut, `True`)

**Renvoie**

représentation sous forme de chaîne du jeu de données

**Type renvoyé**

`str`

**transform\_shapes**(*orig*, *func*, *param*=`None`)

Appliquer la fonction de transformation aux coordonnées de la forme / des annotations du résultat.

**Paramètres**

- **orig** – objet d'origine
- **func** – fonction de transformation
- **param** – paramètre de la fonction de transformation

**update\_metadata\_from**(*other\_metadata* : `dict[str, Any]`) → `None`

Update metadata from another object's metadata (merge result shapes and annotations, and update the rest of the metadata).

**Paramètres**

**other\_metadata** – other object metadata

**update\_metadata\_view\_settings**() → `None`

Mettre à jour les paramètres d'affichage des métadonnées à partir de `Conf.view`

**update\_resultshapes\_from**(*other* : `TypeObj`) → `None`

Mettre à jour la forme géométrique à partir d'un autre objet (fusionner les métadonnées).

**Paramètres**

**other** – autre objet, à partir duquel mettre à jour cet objet

**view**(*parent* : `QWidget` | `None` = `None`, *wordwrap* : `bool` = `True`, *size* : `QSize` | `tuple[int, int]` | `None` = `None`)  
→ `None`

Ouvrir une boîte de dialogue pour afficher le jeu de données

**Paramètres**

- **parent** – widget parent (par défaut, `None` signifie pas de parent)
- **wordwrap** – si `True`, le texte du commentaire est enveloppé
- **size** – taille de la boîte de dialogue (objet `QSize` ou tuple d'entiers (largeur, hauteur))

**write\_config**(*conf* : *UserConfig*, *section* : *str*, *option* : *str*) → *None*

Écrire la configuration dans une instance *UserConfig*

**Paramètres**

- **conf** (*UserConfig*) – instance *UserConfig*
- **section** (*str*) – nom de la section
- **option** (*str*) – nom de l’option

**cdl.obj.read\_image**(*filename* : *str*) → *ImageObj*

Lire une image à partir d’un fichier.

**Paramètres**

**filename** – Nom de fichier.

**Renvoie**

Image.

**cdl.obj.read\_images**(*filename* : *str*) → *list[ImageObj]*

Lire une liste d’images à partir d’un fichier.

**Paramètres**

**filename** – Nom de fichier.

**Renvoie**

Liste d’images.

**cdl.obj.create\_image\_roi**(*geometry* : *Literal*['rectangle', 'circle', 'polygon'], *coords* : *ndarray* | *list[float]* | *list[list[float]]*, *indices* : *bool* = *True*, *singleobj* : *bool* | *None* = *None*, *inverse* : *bool* = *False*, *title* : *str* = "") → *ImageROI*

Create Image Regions of Interest (ROI) object. More ROIs can be added to the object after creation, using the *add\_roi* method.

**Paramètres**

- **geometry** – ROI type (“rectangle”, “circle”, “polygon”)
- **coords** – ROI coords (physical coordinates), [*x0*, *y0*, *dx*, *dy*] for a rectangle, [*xc*, *yc*, *r*] for a circle, or [*x0*, *y0*, *x1*, *y1*, ...] for a polygon (lists or NumPy arrays are accepted). For multiple ROIs, nested lists or NumPy arrays are accepted but with a common geometry type (e.g. [[*xc1*, *yc1*, *r1*], [*xc2*, *yc2*, *r2*], ...] for circles).
- **indices** – if *True*, coordinates are indices, if *False*, they are physical values (default to *True* for images)
- **singleobj** – if *True*, when extracting data defined by ROIs, only one object is created (default to *True*). If *False*, one object is created per single ROI. If *None*, the value is get from the user configuration
- **inverse** – if *True*, ROI is outside the region
- **title** – titre

**Renvoie**

Regions of Interest (ROI) object

**Lève**

**ValueError** – if ROI type is unknown or if the number of coordinates is invalid

**cdl.obj.create\_image**(*title* : *str*, *data* : *ndarray* | *None* = *None*, *metadata* : *dict* | *None* = *None*, *units* : *tuple* | *None* = *None*, *labels* : *tuple* | *None* = *None*) → *ImageObj*

Créer un nouvel objet Image

**Paramètres**

- **title** – titre de l’image
- **data** – données de l’image
- **metadata** – métadonnées de l’image
- **units** – unités X, Y, Z (tuple de chaînes)
- **labels** – étiquettes X, Y, Z (tuple de chaînes)

**Renvoie**

Objet image

```
cdl.obj.create_image_from_param(newparam : NewImageParam, addparam : gds.DataSet | None = None,
                                edit : bool = False, parent : QW.QWidget | None = None) → ImageObj |
                                None
```

Créer un nouvel objet Image à partir d'une boîte de dialogue.

**Paramètres**

- **newparam** – nouveaux paramètres de l'image
- **addparam** – paramètres supplémentaires
- **edit** – Ouvrir une boîte de dialogue pour modifier les paramètres (par défaut : False)
- **parent** – widget parent

**Renvoie**

Nouvel objet image ou None si l'utilisateur a annulé

```
cdl.obj.new_image_param(title : str | None = None, itype : ImageTypes | None = None, height : int | None =
                        None, width : int | None = None, dtype : ImageDatatypes | None = None) →
                        NewImageParam
```

Créer une nouvelle instance de jeu de données Image.

**Paramètres**

- **title** – titre du jeu de données (par défaut : None, utilise le titre par défaut)
- **itype** – type d'image (par défaut : None, utilise le type par défaut)
- **height** – hauteur de l'image (par défaut : None, utilise la hauteur par défaut)
- **width** – largeur de l'image (par défaut : None, utilise la largeur par défaut)
- **dtype** – type de données de l'image (par défaut : None, utilise le type de données par défaut)

**Renvoie**

Nouvelle instance de jeu de données image

```
class cdl.obj.ImageTypes(value, names=None, *, module=None, qualname=None, type=None, start=1,
                          boundary=None)
```

Types d'image

**ZEROS = 'zéros'**

Image remplie de zéros

**EMPTY = 'vide'**

Image vide (remplie de données de l'état de la mémoire)

**GAUSS = 'gaussienne'**

Image gaussienne 2D

**UNIFORMRANDOM = 'aléatoire (loi uniforme)'**

Image remplie de données aléatoires (loi uniforme)

**NORMALRANDOM = 'aléatoire (loi normale)'**

Image remplie de données aléatoires (loi normale)

**class cdl.obj.NewImageParam**

Nouveau jeu de données image

**title**

Titre. Par défaut : None.

**Type**

guidata.dataset.dataitems.StringItem

**height**

Hauteur. Hauteur de l'image (nombre de lignes). Entier supérieur à 1. Par défaut : None.

**Type**

`guidata.dataset.dataitems.IntItem`

**width**

Largeur. Largeur de l'image (nombre de colonnes). Entier supérieur à 1. Par défaut : None.

**Type**

`guidata.dataset.dataitems.IntItem`

**dtype**

Type de données. Sélection unique parmi : `ImageDatatypes.UINT8`, `ImageDatatypes.UINT16`, `ImageDatatypes.INT16`, `ImageDatatypes.FLOAT32`, `ImageDatatypes.FLOAT64`. Par défaut : `ImageDatatypes.UINT8`.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**itype**

Sélection unique parmi : `ImageTypes.ZEROS`, `ImageTypes.EMPTY`, `ImageTypes.GAUSS`, `ImageTypes.UNIFORMRANDOM`, `ImageTypes.NORMALRANDOM`. Par défaut : `ImageTypes.ZEROS`.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**classmethod** `create(title : str, height : int, width : int, dtype : cdl.core.model.image.ImageDatatypes, itype : cdl.core.model.image.ImageTypes) → cdl.core.model.image.NewImageParam`

Renvoie une nouvelle instance de `NewImageParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **title** (*str*) – Titre. Par défaut : None.
- **height** (*int*) – Hauteur. Hauteur de l'image (nombre de lignes). Entier supérieur à 1. Par défaut : None.
- **width** (*int*) – Largeur. Largeur de l'image (nombre de colonnes). Entier supérieur à 1. Par défaut : None.
- **dtype** (`cdl.core.model.image.ImageDatatypes`) – Type de données. Sélection unique parmi : `ImageDatatypes.UINT8`, `ImageDatatypes.UINT16`, `ImageDatatypes.INT16`, `ImageDatatypes.FLOAT32`, `ImageDatatypes.FLOAT64`. Par défaut : `ImageDatatypes.UINT8`.
- **itype** (`cdl.core.model.image.ImageTypes`) – Sélection unique parmi : `ImageTypes.ZEROS`, `ImageTypes.EMPTY`, `ImageTypes.GAUSS`, `ImageTypes.UNIFORMRANDOM`, `ImageTypes.NORMALRANDOM`. Par défaut : `ImageTypes.ZEROS`.

**Renvoie**

Nouvelle instance de `NewImageParam`.

**class** `cdl.obj.Gauss2DParam`

Paramètres gaussiens 2D

**a**

Norm. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**xmin**

Par défaut : -10.

**Type**  
[guidata.dataset.dataitems.FloatItem](#)

**sigma**  
 . Par défaut : 1.0.

**Type**  
[guidata.dataset.dataitems.FloatItem](#)

**xmax**  
 Par défaut : 10.

**Type**  
[guidata.dataset.dataitems.FloatItem](#)

**mu**  
 . Par défaut : 0.0.

**Type**  
[guidata.dataset.dataitems.FloatItem](#)

**ymin**  
 Par défaut : -10.

**Type**  
[guidata.dataset.dataitems.FloatItem](#)

**x0**  
 Par défaut : 0.

**Type**  
[guidata.dataset.dataitems.FloatItem](#)

**ymax**  
 Par défaut : 10.

**Type**  
[guidata.dataset.dataitems.FloatItem](#)

**y0**  
 Par défaut : 0.

**Type**  
[guidata.dataset.dataitems.FloatItem](#)

**classmethod create**(*a : float, xmin : float, sigma : float, xmax : float, mu : float, ymin : float, x0 : float, ymax : float, y0 : float*) → [cdl.core.model.image.Gauss2DParam](#)

Renvoie une nouvelle instance de [Gauss2DParam](#) avec les champs initialisés aux valeurs données.

#### Paramètres

- **a** (*float*) – Norm. Par défaut : None.
- **xmin** (*float*) – Par défaut : -10.
- **sigma** (*float*) – . Par défaut : 1.0.
- **xmax** (*float*) – Par défaut : 10.
- **mu** (*float*) – . Par défaut : 0.0.
- **ymin** (*float*) – Par défaut : -10.
- **x0** (*float*) – Par défaut : 0.
- **ymax** (*float*) – Par défaut : 10.
- **y0** (*float*) – Par défaut : 0.

#### Renvoie

Nouvelle instance de [Gauss2DParam](#).

**class** `cdl.obj.ROI2DParam`

Image ROI parameters

**geometry**

Géométrie. Sélection unique parmi : “rectangle”, “circle”, “polygon”. Par défaut : “rectangle”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**x0**

X<sub>0</sub>. Entier, unité : pixels. Par défaut : None.

**Type**

`guidata.dataset.dataitems.IntItem`

**y0**

Y<sub>0</sub>. Entier, unité : pixels. Par défaut : None.

**Type**

`guidata.dataset.dataitems.IntItem`

**dx**

X. Entier, unité : pixels. Par défaut : None.

**Type**

`guidata.dataset.dataitems.IntItem`

**dy**

Y. Entier, unité : pixels. Par défaut : None.

**Type**

`guidata.dataset.dataitems.IntItem`

**xc**

X<sub>C</sub>. Entier, unité : pixels. Par défaut : None.

**Type**

`guidata.dataset.dataitems.IntItem`

**yc**

Y<sub>C</sub>. Entier, unité : pixels. Par défaut : None.

**Type**

`guidata.dataset.dataitems.IntItem`

**r**

Rayon. Entier, unité : pixels. Par défaut : None.

**Type**

`guidata.dataset.dataitems.IntItem`

**points**

Coordonnées (pixels). Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatArrayItem`

**classmethod** `create(geometry : str, x0 : int, y0 : int, dx : int, dy : int, xc : int, yc : int, r : int, points : numpy.ndarray) → cdl.core.model.image.ROI2DParam`

Renvoie une nouvelle instance de `ROI2DParam` avec les champs initialisés aux valeurs données.

**Paramètres**

— **geometry** (*str*) – Géométrie. Sélection unique parmi : “rectangle”, “circle”, “polygon”. Par défaut : “rectangle”.

- **x0** (*int*) –  $X_0$ . Entier, unité : pixels. Par défaut : None.
- **y0** (*int*) –  $Y_0$ . Entier, unité : pixels. Par défaut : None.
- **dx** (*int*) –  $X$ . Entier, unité : pixels. Par défaut : None.
- **dy** (*int*) –  $Y$ . Entier, unité : pixels. Par défaut : None.
- **xc** (*int*) –  $X_C$ . Entier, unité : pixels. Par défaut : None.
- **yc** (*int*) –  $Y_C$ . Entier, unité : pixels. Par défaut : None.
- **r** (*int*) – Rayon. Entier, unité : pixels. Par défaut : None.
- **points** (*numpy.ndarray*) – Coordonnées (pixels). Par défaut : None.

**Renvoie**

Nouvelle instance de *ROI2DParam*.

**to\_single\_roi**(*obj* : *ImageObj*, *title* : *str* = "") → *PolygonalROI* | *RectangularROI* | *CircularROI*

Convert parameters to single ROI

**Paramètres**

- **obj** – image object (used for conversion of pixel to physical coordinates)
- **title** – ROI title

**Renvoie**

Single ROI

**get\_suffix**() → *str*

Get suffix text representation for ROI extraction

**get\_extracted\_roi**(*obj* : *ImageObj*) → *ImageROI* | *None*

Get extracted ROI, i.e. the remaining ROI after extracting ROI from image.

**Paramètres**

- obj** – image object (used for conversion of pixel to physical coordinates)

When extracting ROIs from an image to multiple images (i.e. one image per ROI), this method returns the ROI that has to be kept in the destination image. This is not necessary for a rectangular ROI : the destination image is simply a crop of the source image according to the ROI coordinates. But for a circular ROI or a polygonal ROI, the destination image is a crop of the source image according to the bounding box of the ROI. Thus, to avoid any loss of information, a ROI has to be defined for the destination image : this is the ROI returned by this method. It's simply the same as the source ROI, but with coordinates adjusted to the destination image. One may called this ROI the « extracted ROI ».

**get\_bounding\_box\_indices**() → *tuple*[*int*, *int*, *int*, *int*]

Get bounding box (pixel coordinates)

**get\_data**(*obj* : *ImageObj*) → *ndarray*

Get data in ROI

**Paramètres**

- obj** – image object

**Renvoie**

Data in ROI

**class** *cdl.obj.ImageROI*(*singleobj* : *bool* | *None* = *None*, *inverse* : *bool* = *False*)

Image Regions of Interest

**Paramètres**

- **singleobj** – if True, when extracting data defined by ROIs, only one object is created (default to True). If False, one object is created per single ROI. If None, the value is get from the user configuration
- **inverse** – if True, ROI is outside the region

**static** **get\_compatible\_single\_roi\_classes**() → *list*[*Type*[*BaseSingleImageROI*]]

Return compatible single ROI classes

**to\_mask**(*obj* : ImageObj) → ndarray[bool]

Create mask from ROI

**Paramètres**

**obj** – image object

**Renvoie**

Mask (boolean array where True values are inside the ROI)

**class** `cdl.obj.ImageDatatypes`(*value*, *names=None*, \*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Types de données d'image

**classmethod** `from_dtype`(*dtype*)

Renvoie le membre à partir du dtype NumPy

**classmethod** `check`()

Vérifie si les types de données sont valides

**UINT8** = 'uint8'

Entier non signé stocké avec 8 bits

**UINT16** = 'uint16'

Entier non signé stocké avec 16 bits

**INT16** = 'int16'

Entier signé stocké avec 16 bits

**FLOAT32** = 'float32'

Nombre flottant stocké avec 32 bits

**FLOAT64** = 'float64'

Nombre flottant stocké avec 64 bits

## 3.4 Computation (cdl.computation)

This package contains the computation functions used by the DataLab project. Those functions operate directly on DataLab objects (i.e. `cdl.obj.SignalObj` and `cdl.obj.ImageObj`) and are designed to be used in the DataLab pipeline, but can be used independently as well.

**Voir aussi :**

The `cdl.computation` package is the main entry point for the DataLab computation functions when manipulating DataLab objects. See the `cdl.algorithms` package for algorithms that operate directly on NumPy arrays.

Each computation module defines a set of computation objects, that is, functions that implement processing features and classes that implement the corresponding parameters (in the form of `guidata.dataset.datatypes.Dataset` subclasses). The computation functions takes a DataLab object (e.g. `cdl.obj.SignalObj`) and a parameter object (e.g. `cdl.param.MovingAverageParam`) as input and return a DataLab object as output (the result of the computation). The parameter object is used to configure the computation function (e.g. the size of the moving average window).

In DataLab overall architecture, the purpose of this package is to provide the computation functions that are used by the `cdl.core.gui.processor` module, based on the algorithms defined in the `cdl.algorithms` module and on the data model defined in the `cdl.obj` (or `cdl.core.model`) module.

The computation modules are organized in subpackages according to their purpose. The following subpackages are available :

- `cdl.computation.base` : Common processing features

- `cdl.computation.signal` : Signal processing features
- `cdl.computation.image` : Image processing features

### 3.4.1 Common processing features

**class** `cdl.computation.base.ArithmeticParam`

Arithmetic parameters

**operator**

Opérateur. Sélection unique parmi : “+”, “-”, “×”, “/”. Par défaut : “+”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**factor**

Facteur. Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**constant**

Constante. Par défaut : 0.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**operation**

Opération. Par défaut : “”.

**Type**

`guidata.dataset.dataitems.StringItem`

**restore\_dtype**

Résultat. Par défaut : True.

**Type**

`guidata.dataset.dataitems.BoolItem`

**classmethod** `create(operator : str, factor : float, constant : float, operation : str, restore_dtype : bool) → cdl.computation.base.ArithmeticParam`

Renvoie une nouvelle instance de `ArithmeticParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **operator** (`str`) – Opérateur. Sélection unique parmi : “+”, “-”, “×”, “/”. Par défaut : “+”.
- **factor** (`float`) – Facteur. Par défaut : 1.0.
- **constant** (`float`) – Constante. Par défaut : 0.0.
- **operation** (`str`) – Opération. Par défaut : “”.
- **restore\_dtype** (`bool`) – Résultat. Par défaut : True.

**Renvoie**

Nouvelle instance de `ArithmeticParam`.

**get\_operation()** → `str`

Return the operation string

**update\_operation(\_item, \_value)**

Update the operation item

**class** `cdl.computation.base.GaussianParam`

Gaussian filter parameters

**sigma**

. Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod create**(*sigma : float*) → *cdl.computation.base.GaussianParam*

Renvoie une nouvelle instance de *GaussianParam* avec les champs initialisés aux valeurs données.

**Paramètres**

**sigma** (*float*) – . Par défaut : 1.0.

**Renvoie**

Nouvelle instance de *GaussianParam*.

**class** *cdl.computation.base.MovingAverageParam*

Moving average parameters

**n**

Taille de la fenêtre glissante. Entier supérieur à 1. Par défaut : 3.

**Type**

`guidata.dataset.dataitems.IntItem`

**mode**

Mode du filtre : - “reflect” : réfléchir les données à la frontière - “constant” : remplir avec une valeur constante - “nearest” : remplir avec la valeur la plus proche - “mirror” : réfléchir les données à la frontière avec les données elles-mêmes - “wrap” : frontière circulaire. Sélection unique parmi : “reflect”, “constant”, “nearest”, “mirror”, “wrap”. Par défaut : “reflect”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**classmethod create**(*n : int, mode : str*) → *cdl.computation.base.MovingAverageParam*

Renvoie une nouvelle instance de *MovingAverageParam* avec les champs initialisés aux valeurs données.

**Paramètres**

- **n** (*int*) – Taille de la fenêtre glissante. Entier supérieur à 1. Par défaut : 3.
- **mode** (*str*) – Mode du filtre : - “reflect” : réfléchir les données à la frontière - “constant” : remplir avec une valeur constante - “nearest” : remplir avec la valeur la plus proche - “mirror” : réfléchir les données à la frontière avec les données elles-mêmes - “wrap” : frontière circulaire. Sélection unique parmi : “reflect”, “constant”, “nearest”, “mirror”, “wrap”. Par défaut : “reflect”.

**Renvoie**

Nouvelle instance de *MovingAverageParam*.

**class** *cdl.computation.base.MovingMedianParam*

Moving median parameters

**n**

Taille de la fenêtre glissante. Entier supérieur à 1, impair. Par défaut : 3.

**Type**

`guidata.dataset.dataitems.IntItem`

**mode**

Mode du filtre : - “reflect” : réfléchir les données à la frontière - “constant” : remplir avec une valeur constante - “nearest” : remplir avec la valeur la plus proche - “mirror” : réfléchir les données à la frontière avec les données elles-mêmes - “wrap” : frontière circulaire. Sélection unique parmi : “reflect”, “constant”, “nearest”, “mirror”, “wrap”. Par défaut : “nearest”.

**Type**`guidata.dataset.dataitems.ChoiceItem`**classmethod** `create(n : int, mode : str) → cdl.computation.base.MovingMedianParam`Renvoie une nouvelle instance de `MovingMedianParam` avec les champs initialisés aux valeurs données.**Paramètres**

- **n** (`int`) – Taille de la fenêtre glissante. Entier supérieur à 1, impair. Par défaut : 3.
- **mode** (`str`) – Mode du filtre : - “reflect” : réfléchir les données à la frontière - “constant” : remplir avec une valeur constante - “nearest” : remplir avec la valeur la plus proche - “mirror” : réfléchir les données à la frontière avec les données elles-mêmes - “wrap” : frontière circulaire. Sélection unique parmi : “reflect”, “constant”, “nearest”, “mirror”, “wrap”. Par défaut : “nearest”.

**Renvoie**Nouvelle instance de `MovingMedianParam`.**class** `cdl.computation.base.ClipParam`

Data clipping parameters

**lower**

Borne inférieure d’écrtage. Par défaut : None.

**Type**`guidata.dataset.dataitems.FloatItem`**upper**

Borne supérieure d’écrtage. Par défaut : None.

**Type**`guidata.dataset.dataitems.FloatItem`**classmethod** `create(lower : float, upper : float) → cdl.computation.base.ClipParam`Renvoie une nouvelle instance de `ClipParam` avec les champs initialisés aux valeurs données.**Paramètres**

- **lower** (`float`) – Borne inférieure d’écrtage. Par défaut : None.
- **upper** (`float`) – Borne supérieure d’écrtage. Par défaut : None.

**Renvoie**Nouvelle instance de `ClipParam`.**class** `cdl.computation.base.NormalizeParam`

Normalize parameters

**method**

Normaliser par rapport à. Sélection unique parmi : “maximum”, “amplitude”, “area”, “energy”, “rms”. Par défaut : “maximum”.

**Type**`guidata.dataset.dataitems.ChoiceItem`**classmethod** `create(method : str) → cdl.computation.base.NormalizeParam`Renvoie une nouvelle instance de `NormalizeParam` avec les champs initialisés aux valeurs données.**Paramètres**

- method** (`str`) – Normaliser par rapport à. Sélection unique parmi : “maximum”, “amplitude”, “area”, “energy”, “rms”. Par défaut : “maximum”.

**Renvoie**Nouvelle instance de `NormalizeParam`.**class** `cdl.computation.base.HistogramParam`

Histogram parameters

**bins**

Nombre de classes. Entier supérieur à 1. Par défaut : 256.

**Type**

`guidata.dataset.dataitems.IntItem`

**lower**

Limite inférieure. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**upper**

Limite supérieure. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(bins : int, lower : float, upper : float) → cdl.computation.base.HistogramParam`

Renvoie une nouvelle instance de *HistogramParam* avec les champs initialisés aux valeurs données.

**Paramètres**

- **bins** (*int*) – Nombre de classes. Entier supérieur à 1. Par défaut : 256.
- **lower** (*float*) – Limite inférieure. Par défaut : None.
- **upper** (*float*) – Limite supérieure. Par défaut : None.

**Renvoie**

Nouvelle instance de *HistogramParam*.

**get\_suffix**(*data : ndarray*) → str

Return suffix for the histogram computation

**Paramètres**

**data** – data array

**class** `cdl.computation.base.FFTParam`

FFT parameters

**shift**

Décaler la fréquence nulle au centre. Par défaut : None.

**Type**

`guidata.dataset.dataitems.BoolItem`

**classmethod** `create(shift : bool) → cdl.computation.base.FFTParam`

Renvoie une nouvelle instance de *FFTParam* avec les champs initialisés aux valeurs données.

**Paramètres**

**shift** (*bool*) – Décaler la fréquence nulle au centre. Par défaut : None.

**Renvoie**

Nouvelle instance de *FFTParam*.

**class** `cdl.computation.base.SpectrumParam`

Spectrum parameters

**log**

Par défaut : False.

**Type**

`guidata.dataset.dataitems.BoolItem`

**classmethod** `create(log : bool) → cdl.computation.base.SpectrumParam`

Renvoie une nouvelle instance de *SpectrumParam* avec les champs initialisés aux valeurs données.

**Paramètres**

**log** (*bool*) – Par défaut : False.

**Renvoie**

Nouvelle instance de *SpectrumParam*.

**class** `cdl.computation.base.ConstantParam`

Parameter used to set a constant value to used in operations

**value**

Constante. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(value : float) → cdl.computation.base.ConstantParam`

Renvoie une nouvelle instance de *ConstantParam* avec les champs initialisés aux valeurs données.

**Paramètres**

**value** (*float*) – Constante. Par défaut : None.

**Renvoie**

Nouvelle instance de *ConstantParam*.

`cdl.computation.base.dst_11(src : SignalObj | ImageObj, name : str, suffix : str | None = None) → SignalObj | ImageObj`

Create a result object, as returned by the callback function of the `cdl.core.gui.processor.base.BaseProcessor.compute_11()` method

**Paramètres**

- **src** – source signal or image object
- **name** – name of the function. If provided, the title of the result object will be `{name}({src.short_id}){suffix}`, unless the name is a single character, in which case the title will be `{src.short_id}{name}{suffix}` where *name* is an operator and *suffix* is the other term of the operation.
- **suffix** – suffix to add to the title. Optional.

**Renvoie**

Result signal or image object

`cdl.computation.base.dst_n1n(src1 : SignalObj | ImageObj, src2 : SignalObj | ImageObj, name : str, suffix : str | None = None) → SignalObj | ImageObj`

Create a result object, as returned by the callback function of the `cdl.core.gui.processor.base.BaseProcessor.compute_n1n()` method

**Paramètres**

- **src1** – input signal or image object
- **src2** – input signal or image object
- **name** – name of the processing function

**Renvoie**

Output signal or image object

`cdl.computation.base.new_signal_result(src : SignalObj | ImageObj, name : str, suffix : str | None = None, units : tuple[str, str] | None = None, labels : tuple[str, str] | None = None) → SignalObj`

Create new signal object as a result of a compute\_11 function

As opposed to the *dst\_11* functions, this function creates a new signal object without copying the original object metadata, except for the « source » entry.

**Paramètres**

- **src** – input signal or image object

- **name** – name of the processing function
- **suffix** – suffix to add to the title
- **units** – units of the output signal
- **labels** – labels of the output signal

**Renvoie**

Output signal object

`cdl.computation.base.calc_resultproperties(title : str, obj : SignalObj | ImageObj, labeledfuncs : dict[str, Callable]) → ResultProperties`

Calculate result properties by executing a computation function on a signal/image object.

**Paramètres**

- **title** – title of the result properties
- **obj** – signal or image object
- **labeledfuncs** – dictionary of labeled computation functions. The keys are the labels of the computation functions and the values are the functions themselves (each function must take a single argument - which is the data of the ROI or the whole signal/image - and return a float)

**Renvoie**

Result properties object

### 3.4.2 Signal processing features

`cdl.computation.signal.restore_data_outside_roi(dst : SignalObj, src : SignalObj) → None`

Restore data outside the Region Of Interest (ROI) of the input signal after a computation, only if the input signal has a ROI, and if the output signal has the same ROI as the input signal, and if the data types are the same, and if the shapes are the same. Otherwise, do nothing.

**Paramètres**

- **dst** – destination signal object
- **src** – source signal object

`class cdl.computation.signal.Wrap11Func(func : Callable, *args : Any, **kwargs : Any)`

Wrap a 1 array → 1 array function (the simple case of  $y1 = f(y0)$ ) to produce a 1 signal → 1 signal function, which can be used inside DataLab's infrastructure to perform computations with `cdl.core.gui.processor.signal.SignalProcessor`.

This wrapping mechanism using a class is necessary for the resulted function to be pickable by the multiprocessing module.

The instance of this wrapper is callable and returns a `cdl.obj.SignalObj` object.

**Exemple**

```
>>> import numpy as np
>>> from cdl.computation.signal import Wrap11Func
>>> import cdl.obj
>>> def square(y):
...     return y**2
>>> compute_square = Wrap11Func(square)
>>> x = np.linspace(0, 10, 100)
>>> y = np.sin(x)
>>> sig0 = cdl.obj.create_signal("Example", x, y)
>>> sig1 = compute_square(sig0)
```

**Paramètres**

- **func** – 1 array  $\rightarrow$  1 array function
- **\*args** – Additional positional arguments to pass to the function
- **\*\*kwargs** – Additional keyword arguments to pass to the function

`cdl.computation.signal.compute_addition(dst : SignalObj, src : SignalObj)  $\rightarrow$  SignalObj`

Add **dst** and **src** signals and return **dst** signal modified in place

**Paramètres**

- **dst** – destination signal
- **src** – source signal

**Renvoie**

Modified **dst** signal (modified in place)

`cdl.computation.signal.compute_product(dst : SignalObj, src : SignalObj)  $\rightarrow$  SignalObj`

Multiply **dst** and **src** signals and return **dst** signal modified in place

**Paramètres**

- **dst** – destination signal
- **src** – source signal

**Renvoie**

Modified **dst** signal (modified in place)

`cdl.computation.signal.compute_addition_constant(src : SignalObj, p : ConstantParam)  $\rightarrow$  SignalObj`

Add **dst** and a constant value and return a the new result signal object

**Paramètres**

- **src** – input signal object
- **p** – constant value

**Renvoie**

Result signal object **src** + **p.value** (new object)

`cdl.computation.signal.compute_difference_constant(src : SignalObj, p : ConstantParam)  $\rightarrow$  SignalObj`

Subtract a constant value from a signal

**Paramètres**

- **src** – input signal object
- **p** – constant value

**Renvoie**

Result signal object **src** - **p.value** (new object)

`cdl.computation.signal.compute_product_constant(src : SignalObj, p : ConstantParam)  $\rightarrow$  SignalObj`

Multiply **dst** by a constant value and return the new result signal object

**Paramètres**

- **src** – input signal object
- **p** – constant value

**Renvoie**

Result signal object **src** \* **p.value** (new object)

`cdl.computation.signal.compute_division_constant(src : SignalObj, p : ConstantParam)  $\rightarrow$  SignalObj`

Divide a signal by a constant value

**Paramètres**

- **src** – input signal object
- **p** – constant value

**Renvoie**

Result signal object **src** / **p.value** (new object)

`cdl.computation.signal.compute_arithmetic`(*src1* : [SignalObj](#), *src2* : [SignalObj](#), *p* : [ArithmeticParam](#)) → [SignalObj](#)

Perform arithmetic operation on two signals

**Paramètres**

- **src1** – source signal 1
- **src2** – source signal 2
- **p** – parameters

**Renvoie**

Result signal object

`cdl.computation.signal.compute_difference`(*src1* : [SignalObj](#), *src2* : [SignalObj](#)) → [SignalObj](#)

Compute difference between two signals

---

**Note :** If uncertainty is available, it is propagated.

---

**Paramètres**

- **src1** – source signal 1
- **src2** – source signal 2

**Renvoie**

Result signal object **src1** - **src2**

`cdl.computation.signal.compute_quadratic_difference`(*src1* : [SignalObj](#), *src2* : [SignalObj](#)) → [SignalObj](#)

Compute quadratic difference between two signals

---

**Note :** If uncertainty is available, it is propagated.

---

**Paramètres**

- **src1** – source signal 1
- **src2** – source signal 2

**Renvoie**

Result signal object (**src1** - **src2**) / sqrt(2.0)

`cdl.computation.signal.compute_division`(*src1* : [SignalObj](#), *src2* : [SignalObj](#)) → [SignalObj](#)

Compute division between two signals

**Paramètres**

- **src1** – source signal 1
- **src2** – source signal 2

**Renvoie**

Result signal object **src1** / **src2**

`cdl.computation.signal.extract_multiple_roi`(*src* : [SignalObj](#), *group* : [DataSetGroup](#)) → [SignalObj](#)

Extract multiple regions of interest from data

**Paramètres**

- **src** – source signal
- **group** – group of parameters

**Renvoie**

Signal with multiple regions of interest

`cdl.computation.signal.extract_single_roi(src : SignalObj, p : ROI1DParam) → SignalObj`

Extract single region of interest from data

**Paramètres**

- **src** – source signal
- **p** – ROI parameters

**Renvoie**

Signal with single region of interest

`cdl.computation.signal.compute_swap_axes(src : SignalObj) → SignalObj`

Swap axes

**Paramètres**

- src** – source signal

**Renvoie**

Result signal object

`cdl.computation.signal.compute_abs(src : SignalObj) → SignalObj`

Compute absolute value with `numpy.absolute`

**Paramètres**

- src** – source signal

**Renvoie**

Result signal object

`cdl.computation.signal.compute_re(src : SignalObj) → SignalObj`

Compute real part with `numpy.real()`

**Paramètres**

- src** – source signal

**Renvoie**

Result signal object

`cdl.computation.signal.compute_im(src : SignalObj) → SignalObj`

Compute imaginary part with `numpy.imag()`

**Paramètres**

- src** – source signal

**Renvoie**

Result signal object

**class** `cdl.computation.signal.DataTypeSParam`

Convert signal data type parameters

**dtype\_str**

Type de données de destination. Type de données de l'image générée. Sélection unique parmi : "float32", "float64", "complex128". Par défaut : "float32".

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**classmethod** `create(dtype_str : str) → cdl.computation.signal.DataTypeSParam`

Renvoie une nouvelle instance de `DataTypeSParam` avec les champs initialisés aux valeurs données.

**Paramètres**

**dtype\_str** (*str*) – Type de données de destination. Type de données de l'image générée. Sélection unique parmi : "float32", "float64", "complex128". Par défaut : "float32".

**Renvoie**

Nouvelle instance de `DataTypeSParam`.

`cdl.computation.signal.compute_astype(src : SignalObj, p : DataTypeSParam) → SignalObj`

Convert data type with `numpy.astype()`

**Paramètres**

- **src** – source signal
- **p** – parameters

**Renvoie**

Result signal object

`cdl.computation.signal.compute_log10(src : SignalObj) → SignalObj`

Compute Log10 with `numpy.log10`

**Paramètres**

- src** – source signal

**Renvoie**

Result signal object

`cdl.computation.signal.compute_exp(src : SignalObj) → SignalObj`

Compute exponential with `numpy.exp`

**Paramètres**

- src** – source signal

**Renvoie**

Result signal object

`cdl.computation.signal.compute_sqrt(src : SignalObj) → SignalObj`

Compute square root with `numpy.sqrt`

**Paramètres**

- src** – source signal

**Renvoie**

Result signal object

**class** `cdl.computation.signal.PowerParam`

Power parameters

**power**

Puissance. Par défaut : 2.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(power : float) → cdl.computation.signal.PowerParam`

Renvoie une nouvelle instance de `PowerParam` avec les champs initialisés aux valeurs données.

**Paramètres**

**power** (*float*) – Puissance. Par défaut : 2.0.

**Renvoie**

Nouvelle instance de `PowerParam`.

`cdl.computation.signal.compute_power(src : SignalObj, p : PowerParam) → SignalObj`

Compute power with `numpy.power`

**Paramètres**

- **src** – source signal
- **p** – parameters

**Renvoie**

Result signal object

**class** `cdl.computation.signal.PeakDetectionParam`

Peak detection parameters

**threshold**

Seuil. Entier compris entre 0 et 100, unité : %. Par défaut : 30.

**Type**

`guidata.dataset.dataitems.IntItem`

**min\_dist**

Distance minimale. Entier supérieur à 1, unité : points. Par défaut : 1.

**Type**

`guidata.dataset.dataitems.IntItem`

**classmethod** `create(threshold : int, min_dist : int) → cdl.computation.signal.PeakDetectionParam`

Renvoie une nouvelle instance de `PeakDetectionParam` avec les champs initialisés aux valeurs données.

**Paramètres**

— **threshold** (`int`) – Seuil. Entier compris entre 0 et 100, unité : %. Par défaut : 30.

— **min\_dist** (`int`) – Distance minimale. Entier supérieur à 1, unité : points. Par défaut : 1.

**Renvoie**

Nouvelle instance de `PeakDetectionParam`.

`cdl.computation.signal.compute_peak_detection(src : SignalObj, p : PeakDetectionParam) → SignalObj`

Peak detection with `cdl.algorithms.signal.peak_indices()`

**Paramètres**

— **src** – source signal

— **p** – parameters

**Renvoie**

Result signal object

`cdl.computation.signal.compute_normalize(src : SignalObj, p : NormalizeParam) → SignalObj`

Normalize data with `cdl.algorithms.signal.normalize()`

**Paramètres**

— **src** – source signal

— **p** – parameters

**Renvoie**

Result signal object

`cdl.computation.signal.compute_derivative(src : SignalObj) → SignalObj`

Compute derivative with `numpy.gradient()`

**Paramètres**

**src** – source signal

**Renvoie**

Result signal object

`cdl.computation.signal.compute_integral(src : SignalObj) → SignalObj`

Compute integral with `scipy.integrate.cumulative_trapezoid()`

**Paramètres**

**src** – source signal

**Renvoie**

Result signal object

**class** `cdl.computation.signal.XYCalibrateParam`

Signal calibration parameters

**axis**

Étalonner. Sélection unique parmi : “x”, “y”. Par défaut : “y”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**a**

Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**b**

Par défaut : 0.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(axis : str, a : float, b : float) → cdl.computation.signal.XYCalibrateParam`

Renvoie une nouvelle instance de `XYCalibrateParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **axis** (*str*) – Étalonner. Sélection unique parmi : “x”, “y”. Par défaut : “y”.
- **a** (*float*) – Par défaut : 1.0.
- **b** (*float*) – Par défaut : 0.0.

**Renvoie**

Nouvelle instance de `XYCalibrateParam`.

`cdl.computation.signal.compute_calibration(src : SignalObj, p : XYCalibrateParam) → SignalObj`

Compute linear calibration

**Paramètres**

- **src** – source signal
- **p** – parameters

**Renvoie**

Result signal object

`cdl.computation.signal.compute_clip(src : SignalObj, p : ClipParam) → SignalObj`

Compute maximum data clipping with `numpy.clip()`

**Paramètres**

- **src** – source signal
- **p** – parameters

**Renvoie**

Result signal object

`cdl.computation.signal.compute_offset_correction(src : SignalObj, p : ROI1DParam) → SignalObj`

Correct offset : subtract the mean value of the signal in the specified range (baseline correction)

**Paramètres**

- **src** – source signal
- **p** – parameters

**Renvoie**

Result signal object

`cdl.computation.signal.compute_gaussian_filter(src : SignalObj, p : GaussianParam) → SignalObj`

Compute gaussian filter with `scipy.ndimage.gaussian_filter()`

**Paramètres**

- **src** – source signal
- **p** – parameters

**Renvoie**

Result signal object

`cdl.computation.signal.compute_moving_average(src : SignalObj, p : MovingAverageParam) → SignalObj`

Compute moving average with `scipy.ndimage.uniform_filter()`

**Paramètres**

- **src** – source signal
- **p** – parameters

**Renvoie**

Result signal object

`cdl.computation.signal.compute_moving_median(src : SignalObj, p : MovingMedianParam) → SignalObj`

Compute moving median with `scipy.ndimage.median_filter()`

**Paramètres**

- **src** – source signal
- **p** – parameters

**Renvoie**

Result signal object

`cdl.computation.signal.compute_wiener(src : SignalObj) → SignalObj`

Compute Wiener filter with `scipy.signal.wiener()`

**Paramètres**

- src** – source signal

**Renvoie**

Result signal object

**class** `cdl.computation.signal.FilterType(value, names=None, *, module=None, qualname=None, type=None, start=1, boundary=None)`

Filter types

**class** `cdl.computation.signal.BaseHighLowBandParam`

Base class for high-pass, low-pass, band-pass and band-stop filters

**method**

Méthode de filtrage. Sélection unique parmi : “bessel”, “butter”, “cheby1”, “cheby2”, “ellip”. Par défaut : “bessel”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**order**

Ordre du filtre. Entier supérieur à 1. Par défaut : 3.

**Type**

`guidata.dataset.dataitems.IntItem`

**f\_cut0**

Fréquence de coupure basse. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**f\_cut1**

Fréquence de coupure haute. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**rp**

Ondulation de la bande passante. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Type**`guidata.dataset.dataitems.FloatItem`**rs**

Atténuation de la bande d'arrêt. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Type**`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(method : str, order : int, f_cut0 : float, f_cut1 : float, rp : float, rs : float) →`  
`cdl.computation.signal.BaseHighLowBandParam`

Renvoie une nouvelle instance de `BaseHighLowBandParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **method** (*str*) – Méthode de filtrage. Sélection unique parmi : “bessel”, “butter”, “cheby1”, “cheby2”, “ellip”. Par défaut : “bessel”.
- **order** (*int*) – Ordre du filtre. Entier supérieur à 1. Par défaut : 3.
- **f\_cut0** (*float*) – Fréquence de coupure basse. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.
- **f\_cut1** (*float*) – Fréquence de coupure haute. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.
- **rp** (*float*) – Ondulation de la bande passante. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.
- **rs** (*float*) – Atténuation de la bande d'arrêt. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Renvoie**Nouvelle instance de `BaseHighLowBandParam`.

**static** `get_nyquist_frequency(obj : SignalObj) → float`

Return the Nyquist frequency of a signal object

**Paramètres****obj** – signal object

**update\_from\_signal** (*obj : SignalObj*) → None

Update the filter parameters from a signal object

**Paramètres****obj** – signal object

**get\_filter\_params** (*obj : SignalObj*) → tuple[float | str, float | str]

Return the filter parameters (a and b) as a tuple. These parameters are used in the `scipy.signal` filter functions (eg. `scipy.signal.filtfilt`).

**Paramètres****obj** – signal object**Renvoie**

filter parameters

**Type renvoyé**`tuple`

**class** `cdl.computation.signal.LowPassFilterParam`

Low-pass filter parameters

**method**

Méthode de filtrage. Sélection unique parmi : “bessel”, “butter”, “cheby1”, “cheby2”, “ellip”. Par défaut : “bessel”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**order**

Ordre du filtre. Entier supérieur à 1. Par défaut : 3.

**Type**

`guidata.dataset.dataitems.IntItem`

**f\_cut0**

Fréquence de coupure basse. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**f\_cut1**

Fréquence de coupure haute. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**rp**

Ondulation de la bande passante. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Type**

`guidata.dataset.dataitems.FloatItem`

**rs**

Atténuation de la bande d’arrêt. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** **create**(*method* : *str*, *order* : *int*, *f\_cut0* : *float*, *f\_cut1* : *float*, *rp* : *float*, *rs* : *float*) → *cdl.computation.signal.LowPassFilterParam*

Renvoie une nouvelle instance de *LowPassFilterParam* avec les champs initialisés aux valeurs données.

**Paramètres**

- **method** (*str*) – Méthode de filtrage. Sélection unique parmi : “bessel”, “butter”, “cheby1”, “cheby2”, “ellip”. Par défaut : “bessel”.
- **order** (*int*) – Ordre du filtre. Entier supérieur à 1. Par défaut : 3.
- **f\_cut0** (*float*) – Fréquence de coupure basse. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.
- **f\_cut1** (*float*) – Fréquence de coupure haute. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.
- **rp** (*float*) – Ondulation de la bande passante. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.
- **rs** (*float*) – Atténuation de la bande d’arrêt. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Renvoie**

Nouvelle instance de *LowPassFilterParam*.

**class** `cdl.computation.signal.HighPassFilterParam`

High-pass filter parameters

**method**

Méthode de filtrage. Sélection unique parmi : “bessel”, “butter”, “cheby1”, “cheby2”, “ellip”. Par défaut : “bessel”.

**Type**`guidata.dataset.dataitems.ChoiceItem`**order**

Ordre du filtre. Entier supérieur à 1. Par défaut : 3.

**Type**`guidata.dataset.dataitems.IntItem`**f\_cut0**

Fréquence de coupure basse. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.

**Type**`guidata.dataset.dataitems.FloatItem`**f\_cut1**

Fréquence de coupure haute. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.

**Type**`guidata.dataset.dataitems.FloatItem`**rp**

Ondulation de la bande passante. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Type**`guidata.dataset.dataitems.FloatItem`**rs**

Atténuation de la bande d'arrêt. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Type**`guidata.dataset.dataitems.FloatItem`

**classmethod create**(*method* : *str*, *order* : *int*, *f\_cut0* : *float*, *f\_cut1* : *float*, *rp* : *float*, *rs* : *float*) → *cdl.computation.signal.HighPassFilterParam*

Renvoie une nouvelle instance de *HighPassFilterParam* avec les champs initialisés aux valeurs données.**Paramètres**

- **method** (*str*) – Méthode de filtrage. Sélection unique parmi : “bessel”, “butter”, “cheby1”, “cheby2”, “ellip”. Par défaut : “bessel”.
- **order** (*int*) – Ordre du filtre. Entier supérieur à 1. Par défaut : 3.
- **f\_cut0** (*float*) – Fréquence de coupure basse. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.
- **f\_cut1** (*float*) – Fréquence de coupure haute. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.
- **rp** (*float*) – Ondulation de la bande passante. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.
- **rs** (*float*) – Atténuation de la bande d'arrêt. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Renvoie**Nouvelle instance de *HighPassFilterParam*.

**class** `cdl.computation.signal.BandPassFilterParam`

Band-pass filter parameters

**method**

Méthode de filtrage. Sélection unique parmi : “bessel”, “butter”, “cheby1”, “cheby2”, “ellip”. Par défaut : “bessel”.

**Type**`guidata.dataset.dataitems.ChoiceItem`

**order**

Ordre du filtre. Entier supérieur à 1. Par défaut : 3.

**Type**

`guidata.dataset.dataitems.IntItem`

**f\_cut0**

Fréquence de coupure basse. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**f\_cut1**

Fréquence de coupure haute. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**rp**

Ondulation de la bande passante. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Type**

`guidata.dataset.dataitems.FloatItem`

**rs**

Atténuation de la bande d'arrêt. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(method : str, order : int, f_cut0 : float, f_cut1 : float, rp : float, rs : float) →`  
`cdl.computation.signal.BandPassFilterParam`

Renvoie une nouvelle instance de `BandPassFilterParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **method** (*str*) – Méthode de filtrage. Sélection unique parmi : “bessel”, “butter”, “cheby1”, “cheby2”, “ellip”. Par défaut : “bessel”.
- **order** (*int*) – Ordre du filtre. Entier supérieur à 1. Par défaut : 3.
- **f\_cut0** (*float*) – Fréquence de coupure basse. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.
- **f\_cut1** (*float*) – Fréquence de coupure haute. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.
- **rp** (*float*) – Ondulation de la bande passante. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.
- **rs** (*float*) – Atténuation de la bande d'arrêt. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Renvoie**

Nouvelle instance de `BandPassFilterParam`.

**class** `cdl.computation.signal.BandStopFilterParam`

Band-stop filter parameters

**method**

Méthode de filtrage. Sélection unique parmi : “bessel”, “butter”, “cheby1”, “cheby2”, “ellip”. Par défaut : “bessel”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**order**

Ordre du filtre. Entier supérieur à 1. Par défaut : 3.

**Type**`guidata.dataset.dataitems.IntItem`**f\_cut0**

Fréquence de coupure basse. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.

**Type**`guidata.dataset.dataitems.FloatItem`**f\_cut1**

Fréquence de coupure haute. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.

**Type**`guidata.dataset.dataitems.FloatItem`**rp**

Ondulation de la bande passante. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Type**`guidata.dataset.dataitems.FloatItem`**rs**

Atténuation de la bande d'arrêt. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Type**`guidata.dataset.dataitems.FloatItem`

**classmethod create**(*method* : *str*, *order* : *int*, *f\_cut0* : *float*, *f\_cut1* : *float*, *rp* : *float*, *rs* : *float*) → *cdl.computation.signal.BandStopFilterParam*

Renvoie une nouvelle instance de *BandStopFilterParam* avec les champs initialisés aux valeurs données.**Paramètres**

- **method** (*str*) – Méthode de filtrage. Sélection unique parmi : “bessel”, “butter”, “cheby1”, “cheby2”, “ellip”. Par défaut : “bessel”.
- **order** (*int*) – Ordre du filtre. Entier supérieur à 1. Par défaut : 3.
- **f\_cut0** (*float*) – Fréquence de coupure basse. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.
- **f\_cut1** (*float*) – Fréquence de coupure haute. Flottant supérieur à 0, non nul, unité : hz. Par défaut : None.
- **rp** (*float*) – Ondulation de la bande passante. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.
- **rs** (*float*) – Atténuation de la bande d'arrêt. Flottant supérieur à 0, non nul, unité : db. Par défaut : 1.

**Renvoie**Nouvelle instance de *BandStopFilterParam*.

`cdl.computation.signal.compute_filter`(*src* : *SignalObj*, *p* : *BaseHighLowBandParam*) → *SignalObj*

Compute frequency filter (low-pass, high-pass, band-pass, band-stop), with `scipy.signal.filtfilt()`**Paramètres**

- **src** – source signal
- **p** – parameters

**Renvoie**

Result signal object

`cdl.computation.signal.compute_fft`(*src* : *SignalObj*, *p* : *FFTParam* | *None* = *None*) → *SignalObj*

Compute FFT with `cdl.algorithms.signal.fft1d()`**Paramètres**

- **src** – source signal
- **p** – parameters

**Renvoie**  
Result signal object

`cdl.computation.signal.compute_ifft(src : SignalObj, p : FFTParam | None = None) → SignalObj`  
Compute iFFT with `cdl.algorithms.signal.ifft1d()`

**Paramètres**  
— **src** – source signal  
— **p** – parameters

**Renvoie**  
Result signal object

`cdl.computation.signal.compute_magnitude_spectrum(src : SignalObj, p : SpectrumParam | None = None) → SignalObj`  
Compute magnitude spectrum with `cdl.algorithms.signal.magnitude_spectrum()`

**Paramètres**  
— **src** – source signal  
— **p** – parameters

**Renvoie**  
Result signal object

`cdl.computation.signal.compute_phase_spectrum(src : SignalObj) → SignalObj`  
Compute phase spectrum with `cdl.algorithms.signal.phase_spectrum()`

**Paramètres**  
**src** – source signal

**Renvoie**  
Result signal object

`cdl.computation.signal.compute_psd(src : SignalObj, p : SpectrumParam | None = None) → SignalObj`  
Compute power spectral density with `cdl.algorithms.signal.psd()`

**Paramètres**  
— **src** – source signal  
— **p** – parameters

**Renvoie**  
Result signal object

**class** `cdl.computation.signal.PolynomialFitParam`  
Polynomial fitting parameters

**degree**  
Degré. Entier compris entre 1 et 10. Par défaut : 3.

**Type**  
`guidata.dataset.dataitems.IntItem`

**classmethod** `create(degree : int) → cdl.computation.signal.PolynomialFitParam`  
Renvoie une nouvelle instance de `PolynomialFitParam` avec les champs initialisés aux valeurs données.

**Paramètres**  
**degree** (`int`) – Degré. Entier compris entre 1 et 10. Par défaut : 3.

**Renvoie**  
Nouvelle instance de `PolynomialFitParam`.

`cdl.computation.signal.compute_histogram(src : SignalObj, p : HistogramParam) → SignalObj`  
Compute histogram with `numpy.histogram()`

**Paramètres**

- **src** – source signal
- **p** – parameters

**Renvoie**

Result signal object

**class** `cdl.computation.signal.InterpolationParam`

Interpolation parameters

**method**

Méthode d'interpolation. Sélection unique parmi : “linear”, “spline”, “quadratic”, “cubic”, “barycentric”, “pchip”. Par défaut : “linear”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**fill\_value**

Valeur de remplissage. Valeur à utiliser pour les points en dehors du domaine d'interpolation (utilisé uniquement avec les méthodes linéaire, cubique et pchip). Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(method : str, fill_value : float) → cdl.computation.signal.InterpolationParam`

Renvoie une nouvelle instance de `InterpolationParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **method** (*str*) – Méthode d'interpolation. Sélection unique parmi : “linear”, “spline”, “quadratic”, “cubic”, “barycentric”, “pchip”. Par défaut : “linear”.
- **fill\_value** (*float*) – Valeur de remplissage. Valeur à utiliser pour les points en dehors du domaine d'interpolation (utilisé uniquement avec les méthodes linéaire, cubique et pchip). Par défaut : None.

**Renvoie**

Nouvelle instance de `InterpolationParam`.

`cdl.computation.signal.compute_interpolation(src1 : SignalObj, src2 : SignalObj, p : InterpolationParam) → SignalObj`

Interpolate data with `cdl.algorithms.signal.interpolate()`

**Paramètres**

- **src1** – source signal 1
- **src2** – source signal 2
- **p** – parameters

**Renvoie**

Result signal object

**class** `cdl.computation.signal.ResamplingParam`

Resample parameters

**method**

Méthode d'interpolation. Sélection unique parmi : “linear”, “spline”, “quadratic”, “cubic”, “barycentric”, “pchip”. Par défaut : “linear”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**fill\_value**

Valeur de remplissage. Valeur à utiliser pour les points en dehors du domaine d'interpolation (utilisé uniquement avec les méthodes linéaire, cubique et pchip). Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**xmin** $X_{\min}$ . Par défaut : None.**Type**

guidata.dataset.dataitems.FloatItem

**xmax** $X_{\max}$ . Par défaut : None.**Type**

guidata.dataset.dataitems.FloatItem

**mode**

Sélection unique parmi : “dx”, “nbpts”. Par défaut : “nbpts”.

**Type**

guidata.dataset.dataitems.ChoiceItem

**dx**

X. Par défaut : None.

**Type**

guidata.dataset.dataitems.FloatItem

**nbpts**

Nombre de points. Par défaut : None.

**Type**

guidata.dataset.dataitems.IntItem

**classmethod create**(*method* : str, *fill\_value* : float, *xmin* : float, *xmax* : float, *mode* : str, *dx* : float, *nbpts* : int) → *cdl.computation.signal.ResamplingParam*

Renvoie une nouvelle instance de *ResamplingParam* avec les champs initialisés aux valeurs données.**Paramètres**

- **method** (str) – Méthode d’interpolation. Sélection unique parmi : “linear”, “spline”, “quadratic”, “cubic”, “barycentric”, “pchip”. Par défaut : “linear”.
- **fill\_value** (float) – Valeur de remplissage. Valeur à utiliser pour les points en dehors du domaine d’interpolation (utilisé uniquement avec les méthodes linéaire, cubique et pchip). Par défaut : None.
- **xmin** (float) –  $X_{\min}$ . Par défaut : None.
- **xmax** (float) –  $X_{\max}$ . Par défaut : None.
- **mode** (str) – Sélection unique parmi : “dx”, “nbpts”. Par défaut : “nbpts”.
- **dx** (float) – X. Par défaut : None.
- **nbpts** (int) – Nombre de points. Par défaut : None.

**Renvoie**Nouvelle instance de *ResamplingParam*.

*cdl.computation.signal.compute\_resampling*(*src* : SignalObj, *p* : ResamplingParam) → *SignalObj*

Resample data with *cdl.algorithms.signal.interpolate()***Paramètres**

- **src** – source signal
- **p** – parameters

**Renvoie**

Result signal object

**class** *cdl.computation.signal.DetrendingParam*

Detrending parameters

**method**

Méthode d'élimination de la tendance. Sélection unique parmi : "linear", "constant". Par défaut : "linear".

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**classmethod** `create(method : str) → cdl.computation.signal.DetrendingParam`

Renvoie une nouvelle instance de `DetrendingParam` avec les champs initialisés aux valeurs données.

**Paramètres**

**method** (*str*) – Méthode d'élimination de la tendance. Sélection unique parmi : "linear", "constant". Par défaut : "linear".

**Renvoie**

Nouvelle instance de `DetrendingParam`.

`cdl.computation.signal.compute_detrending(src : SignalObj, p : DetrendingParam) → SignalObj`

Detrend data with `scipy.signal.detrend()`

**Paramètres**

- **src** – source signal
- **p** – parameters

**Renvoie**

Result signal object

`cdl.computation.signal.compute_convolution(src1 : SignalObj, src2 : SignalObj) → SignalObj`

Compute convolution of two signals with `scipy.signal.convolve()`

**Paramètres**

- **src1** – source signal 1
- **src2** – source signal 2

**Renvoie**

Result signal object

**class** `cdl.computation.signal.WindowingParam`

Windowing parameters

**method**

Méthode. Sélection unique parmi : "barthann", "bartlett", "blackman", "blackman-harris", "bohman", "boxcar", "cosine", "exponential", "flat-top", "gaussian", "hamming", "hanning", "kaiser", "lanczos", "nuttall", "parzen", "rectangular", "taylor", "tukey". Par défaut : "hamming".

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**alpha**

Paramètre de forme de la fonction de fenêtrage de tukey. Par défaut : 0.5.

**Type**

`guidata.dataset.dataitems.FloatItem`

**beta**

Paramètre de forme de la fonction de fenêtrage de kaiser. Par défaut : 14.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**sigma**

Paramètre de forme de la fonction de fenêtrage gaussienne. Par défaut : 0.5.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(method : str, alpha : float, beta : float, sigma : float) → cdl.computation.signal.WindowingParam`

Renvoie une nouvelle instance de `WindowingParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **method** (*str*) – Méthode. Sélection unique parmi : “barthann”, “bartlett”, “blackman”, “blackman-harris”, “bohman”, “boxcar”, “cosine”, “exponential”, “flat-top”, “gaussian”, “hamming”, “hanning”, “kaiser”, “lanczos”, “nuttall”, “parzen”, “rectangular”, “taylor”, “tukey”. Par défaut : “hamming”.
- **alpha** (*float*) – Paramètre de forme de la fonction de fenêtrage de tukey. Par défaut : 0.5.
- **beta** (*float*) – Paramètre de forme de la fonction de fenêtrage de kaiser. Par défaut : 14.0.
- **sigma** (*float*) – Paramètre de forme de la fonction de fenêtrage gaussienne. Par défaut : 0.5.

**Renvoie**

Nouvelle instance de `WindowingParam`.

`cdl.computation.signal.compute_windowing(src : SignalObj, p : WindowingParam) → SignalObj`

Compute windowing (available methods : hamming, hanning, bartlett, blackman, tukey, rectangular) with `cdl.algorithms.signal.windowing()`

**Paramètres**

- **dst** – destination signal
- **src** – source signal

**Renvoie**

Result signal object

`cdl.computation.signal.compute_reverse_x(src : SignalObj) → SignalObj`

Reverse x-axis

**Paramètres**

- src** – source signal

**Renvoie**

Result signal object

`cdl.computation.signal.calc_resultshape(title : str, shape : Literal['rectangle', 'circle', 'ellipse', 'segment', 'marker', 'point', 'polygon'], obj : SignalObj, func : Callable, *args : Any, add_label : bool = False) → ResultShape | None`

Calculate result shape by executing a computation function on a signal object, taking into account the signal ROIs.

**Paramètres**

- **title** – result title
- **shape** – result shape kind
- **obj** – input image object
- **func** – computation function
- **\*args** – computation function arguments
- **add\_label** – if True, add a label item (and the geometrical shape) to plot (default to False)

**Renvoie**

Result shape object or None if no result is found

**Avertissement :** The computation function must take either a single argument (the data) or multiple arguments (the data followed by the computation parameters).

Moreover, the computation function must return a 1D NumPy array (or a list, or a tuple) containing the result of the computation.

**class** `cdl.computation.signal.FWHMParam`

FWHM parameters

**method**

Méthode. Sélection unique parmi : “zero-crossing”, “gauss”, “lorentz”, “voigt”. Par défaut : “zero-crossing”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**xmin**

$X_{\text{MIN}}$ . Borne x inférieure (vide pour aucune limite, c’est-à-dire le début du signal). Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**xmax**

$X_{\text{MAX}}$ . Borne x supérieure (vide pour aucune limite, c’est-à-dire la fin du signal). Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(method : str, xmin : float, xmax : float) → cdl.computation.signal.FWHMParam`

Renvoie une nouvelle instance de *FWHMParam* avec les champs initialisés aux valeurs données.

**Paramètres**

- **method** (*str*) – Méthode. Sélection unique parmi : “zero-crossing”, “gauss”, “lorentz”, “voigt”. Par défaut : “zero-crossing”.
- **xmin** (*float*) –  $X_{\text{MIN}}$ . Borne x inférieure (vide pour aucune limite, c’est-à-dire le début du signal). Par défaut : None.
- **xmax** (*float*) –  $X_{\text{MAX}}$ . Borne x supérieure (vide pour aucune limite, c’est-à-dire la fin du signal). Par défaut : None.

**Renvoie**

Nouvelle instance de *FWHMParam*.

`cdl.computation.signal.compute_fwhm(obj : SignalObj, param : FWHMParam) → ResultShape | None`

Compute FWHM with `cdl.algorithms.signal.fwhm()`

**Paramètres**

- **obj** – source signal
- **param** – parameters

**Renvoie**

Segment coordinates

`cdl.computation.signal.compute_fw1e2(obj : SignalObj) → ResultShape | None`

Compute FW at  $1/e^2$  with `cdl.algorithms.signal.fw1e2()`

**Paramètres**

- obj** – source signal

**Renvoie**

Segment coordinates

`cdl.computation.signal.compute_stats(obj : SignalObj) → ResultProperties`

Compute statistics on a signal

**Paramètres**

- obj** – source signal

**Renvoie**

Result properties object

`cdl.computation.signal.compute_bandwidth_3db(obj: SignalObj) → ResultProperties`

Compute bandwidth at -3 dB with `cdl.algorithms.signal.bandwidth()`

**Paramètres**

**obj** – source signal

**Renvoie**

Result properties with bandwidth

**class** `cdl.computation.signal.DynamicParam`

Parameters for dynamic range computation (ENOB, SNR, SINAD, THD, SFDR)

**full\_scale**

Pleine échelle. Flottant supérieur à 0.0, unité : v. Par défaut : 0.16.

**Type**

`guidata.dataset.dataitems.FloatItem`

**unit**

Unité. Unité pour sinad. Sélection unique parmi : “dBc”, “dBFS”. Par défaut : “dBc”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**nb\_harm**

Nombre d’harmoniques. Nombre d’harmoniques à considérer pour le thd. Entier supérieur à 1. Par défaut : 5.

**Type**

`guidata.dataset.dataitems.IntItem`

**classmethod** `create(full_scale: float, unit: str, nb_harm: int) → cdl.computation.signal.DynamicParam`

Renvoie une nouvelle instance de `DynamicParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **full\_scale** (`float`) – Pleine échelle. Flottant supérieur à 0.0, unité : v. Par défaut : 0.16.
- **unit** (`str`) – Unité. Unité pour sinad. Sélection unique parmi : “dBc”, “dBFS”. Par défaut : “dBc”.
- **nb\_harm** (`int`) – Nombre d’harmoniques. Nombre d’harmoniques à considérer pour le thd. Entier supérieur à 1. Par défaut : 5.

**Renvoie**

Nouvelle instance de `DynamicParam`.

`cdl.computation.signal.compute_dynamic_parameters(src: SignalObj, p: DynamicParam) → ResultProperties`

Compute Dynamic parameters using the following functions :

- Freq : `cdl.algorithms.signal.sinus_frequency()`
- ENOB : `cdl.algorithms.signal.enob()`
- SNR : `cdl.algorithms.signal.snr()`
- SINAD : `cdl.algorithms.signal.sinad()`
- THD : `cdl.algorithms.signal.thd()`
- SFDR : `cdl.algorithms.signal.sfdr()`

**Paramètres**

- **src** – source signal
- **p** – parameters

**Renvoie**

Result properties with ENOB, SNR, SINAD, THD, SFDR

`cdl.computation.signal.compute_sampling_rate_period(obj : SignalObj) → ResultProperties`

Compute sampling rate and period using the following functions :

- `fs : cdl.algorithms.signal.sampling_rate()`
- `T : cdl.algorithms.signal.sampling_period()`

**Paramètres**

**obj** – source signal

**Renvoie**

Result properties with sampling rate and period

`cdl.computation.signal.compute_contrast(obj : SignalObj) → ResultProperties`

Compute contrast with `cdl.algorithms.signal.contrast()`

`cdl.computation.signal.compute_x_at_minmax(obj : SignalObj) → ResultProperties`

Compute x at min/max

### 3.4.3 Image processing features

#### Base image processing features

`cdl.computation.image.restore_data_outside_roi(dst : ImageObj, src : ImageObj) → None`

Restore data outside the Region Of Interest (ROI) of the input image after a computation, only if the input image has a ROI, and if the output image has the same ROI as the input image, and if the data types are compatible, and if the shapes are the same. Otherwise, do nothing.

**Paramètres**

- **dst** – output image object
- **src** – input image object

**class** `cdl.computation.image.Wrap11Func(func : Callable, *args : Any, **kwargs : Any)`

Wrap a 1 array → 1 array function to produce a 1 image → 1 image function, which can be used inside DataLab's infrastructure to perform computations with `cdl.core.gui.processor.image.ImageProcessor`.

This wrapping mechanism using a class is necessary for the resulted function to be pickable by the multiprocessing module.

The instance of this wrapper is callable and returns a `cdl.obj.ImageObj` object.

#### Exemple

```
>>> import numpy as np
>>> from cdl.computation.image import Wrap11Func
>>> import cdl.obj
>>> def add_noise(data):
...     return data + np.random.random(data.shape)
>>> compute_add_noise = Wrap11Func(add_noise)
>>> data= np.ones((100, 100))
>>> ima0 = cdl.obj.create_image("Example", data)
>>> ima1 = compute_add_noise(ima0)
```

**Paramètres**

- **func** – 1 array → 1 array function
- **\*args** – Additional positional arguments to pass to the function

— **\*\*kwargs** – Additional keyword arguments to pass to the function

`cdl.computation.image.dst_11_signal(src : ImageObj, name : str, suffix : str | None = None) → SignalObj`

Create a result signal object, as returned by the callback function of the `cdl.core.gui.processor.base.BaseProcessor.compute_11()` method

**Paramètres**

— **src** – input image object  
— **name** – name of the processing function

**Renvoie**

Output signal object

`cdl.computation.image.compute_addition(dst : ImageObj, src : ImageObj) → ImageObj`

Add **dst** and **src** images and return **dst** image modified in place

**Paramètres**

— **dst** – output image object  
— **src** – input image object

**Renvoie**

Output image object (modified in place)

`cdl.computation.image.compute_product(dst : ImageObj, src : ImageObj) → ImageObj`

Multiply **dst** and **src** images and return **dst** image modified in place

**Paramètres**

— **dst** – output image object  
— **src** – input image object

**Renvoie**

Output image object (modified in place)

`cdl.computation.image.compute_addition_constant(src : ImageObj, p : ConstantParam) → ImageObj`

Add **dst** and a constant value and return the new result image object

**Paramètres**

— **src** – input image object  
— **p** – constant value

**Renvoie**

Result image object **src + p.value** (new object)

`cdl.computation.image.compute_difference_constant(src : ImageObj, p : ConstantParam) → ImageObj`

Subtract a constant value from an image and return the new result image object

**Paramètres**

— **src** – input image object  
— **p** – constant value

**Renvoie**

Result image object **src - p.value** (new object)

`cdl.computation.image.compute_product_constant(src : ImageObj, p : ConstantParam) → ImageObj`

Multiply **dst** by a constant value and return the new result image object

**Paramètres**

— **src** – input image object  
— **p** – constant value

**Renvoie**

Result image object **src \* p.value** (new object)

`cdl.computation.image.compute_division_constant(src : ImageObj, p : ConstantParam) → ImageObj`

Divide an image by a constant value and return the new result image object

**Paramètres**

- **src** – input image object
- **p** – constant value

**Renvoie**

Result image object **src** / **p.value** (new object)

`cdl.computation.image.compute_arithmetic(src1 : ImageObj, src2 : ImageObj, p : ArithmeticParam) → ImageObj`

Compute arithmetic operation on two images

**Paramètres**

- **src1** – input image object
- **src2** – input image object
- **p** – arithmetic parameters

**Renvoie**

Result image object

`cdl.computation.image.compute_difference(src1 : ImageObj, src2 : ImageObj) → ImageObj`

Compute difference between two images

**Paramètres**

- **src1** – input image object
- **src2** – input image object

**Renvoie**

Result image object **src1** - **src2** (new object)

`cdl.computation.image.compute_quadratic_difference(src1 : ImageObj, src2 : ImageObj) → ImageObj`

Compute quadratic difference between two images

**Paramètres**

- **src1** – input image object
- **src2** – input image object

**Renvoie**

Result image object (**src1** - **src2**) / sqrt(2.0) (new object)

`cdl.computation.image.compute_division(src1 : ImageObj, src2 : ImageObj) → ImageObj`

Compute division between two images

**Paramètres**

- **src1** – input image object
- **src2** – input image object

**Renvoie**

Result image object **src1** / **src2** (new object)

**class** `cdl.computation.image.FlatFieldParam`

Flat-field parameters

**threshold**

Seuil. Par défaut : 0.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(threshold : float) → cdl.computation.image.FlatFieldParam`

Renvoie une nouvelle instance de `FlatFieldParam` avec les champs initialisés aux valeurs données.

**Paramètres****threshold** (*float*) – Seuil. Par défaut : 0.0.**Renvoie**Nouvelle instance de *FlatFieldParam*.

```
cdl.computation.image.compute_flatfield(src1 : ImageObj, src2 : ImageObj, p : FlatFieldParam) →
    ImageObj
```

Compute flat field correction with *cdl.algorithms.image.flatfield()***Paramètres**

- **src1** – raw data image object
- **src2** – flat field image object
- **p** – flat field parameters

**Renvoie**

Output image object

```
cdl.computation.image.compute_normalize(src : ImageObj, p : NormalizeParam) → ImageObj
```

Normalize image data depending on its maximum, with *cdl.algorithms.image.normalize()***Paramètres****src** – input image object**Renvoie**

Output image object

```
class cdl.computation.image.LogP1Param
```

Log10 parameters

**n**

Par défaut : None.

**Type***guidata.dataset.dataitems.FloatItem*

```
classmethod create(n : float) → cdl.computation.image.LogP1Param
```

Renvoie une nouvelle instance de *LogP1Param* avec les champs initialisés aux valeurs données.**Paramètres****n** (*float*) – Par défaut : None.**Renvoie**Nouvelle instance de *LogP1Param*.

```
cdl.computation.image.compute_logp1(src : ImageObj, p : LogP1Param) → ImageObj
```

Compute log10(z+n) with *numpy.log10***Paramètres**

- **src** – input image object
- **p** – parameters

**Renvoie**

Output image object

```
class cdl.computation.image.RotateParam
```

Rotate parameters

**angle**

Angle (°). Par défaut : None.

**Type***guidata.dataset.dataitems.FloatItem*

**mode**

Sélection unique parmi : “constant”, “nearest”, “reflect”, “wrap”. Par défaut : “constant”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**cval**

Valeur utilisée pour les points situés en dehors des frontières de l’image d’origine (si le mode est “constant”).  
Par défaut : 0.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**reshape**

Redimensionner l’image de destination de sorte qu’elle puisse contenir la totalité de l’image source. Par défaut : False.

**Type**

`guidata.dataset.dataitems.BoolItem`

**prefilter**

Par défaut : True.

**Type**

`guidata.dataset.dataitems.BoolItem`

**order**

Ordre. Ordre de l’interpolation de type spline. Entier compris entre 0 et 5. Par défaut : 3.

**Type**

`guidata.dataset.dataitems.IntItem`

**classmethod** **create**(*angle : float, mode : str, cval : float, reshape : bool, prefilter : bool, order : int*) → *cdl.computation.image.RotateParam*

Renvoie une nouvelle instance de *[RotateParam](#)* avec les champs initialisés aux valeurs données.

**Paramètres**

- **angle** (*float*) – Angle (°). Par défaut : None.
- **mode** (*str*) – Sélection unique parmi : “constant”, “nearest”, “reflect”, “wrap”. Par défaut : “constant”.
- **cval** (*float*) – Valeur utilisée pour les points situés en dehors des frontières de l’image d’origine (si le mode est “constant”). Par défaut : 0.0.
- **reshape** (*bool*) – Redimensionner l’image de destination de sorte qu’elle puisse contenir la totalité de l’image source. Par défaut : False.
- **prefilter** (*bool*) – Par défaut : True.
- **order** (*int*) – Ordre. Ordre de l’interpolation de type spline. Entier compris entre 0 et 5. Par défaut : 3.

**Renvoie**

Nouvelle instance de *[RotateParam](#)*.

`cdl.computation.image.rotate_obj_coords`(*angle : float, obj : ImageObj, orig : ImageObj, coords : ndarray*) → None

Apply rotation to coords associated to image obj

**Paramètres**

- **angle** – rotation angle (in degrees)
- **obj** – image object
- **orig** – original image object
- **coords** – coordinates to rotate

**Renvoie**

Output data

`cdl.computation.image.rotate_obj_alpha(obj : ImageObj, orig : ImageObj, coords : ndarray, p : RotateParam) → None`

Apply rotation to coords associated to image obj

`cdl.computation.image.compute_rotate(src : ImageObj, p : RotateParam) → ImageObj`

Rotate data with `scipy.ndimage.rotate()`

**Paramètres**

- **src** – input image object
- **p** – parameters

**Renvois**

Output image object

`cdl.computation.image.rotate_obj_90(dst : ImageObj, src : ImageObj, coords : ndarray) → None`

Apply rotation to coords associated to image obj

`cdl.computation.image.compute_rotate90(src : ImageObj) → ImageObj`

Rotate data 90° with `numpy.rot90()`

**Paramètres**

- src** – input image object

**Renvois**

Output image object

`cdl.computation.image.rotate_obj_270(dst : ImageObj, src : ImageObj, coords : ndarray) → None`

Apply rotation to coords associated to image obj

`cdl.computation.image.compute_rotate270(src : ImageObj) → ImageObj`

Rotate data 270° with `numpy.rot90()`

**Paramètres**

- src** – input image object

**Renvois**

Output image object

`cdl.computation.image.hflip_coords(dst : ImageObj, src : ImageObj, coords : ndarray) → None`

Apply HFlip to coords

`cdl.computation.image.compute_fliph(src : ImageObj) → ImageObj`

Flip data horizontally with `numpy.fliplr()`

**Paramètres**

- src** – input image object

**Renvois**

Output image object

`cdl.computation.image.vflip_coords(dst : ImageObj, src : ImageObj, coords : ndarray) → None`

Apply VFlip to coords

`cdl.computation.image.compute_flipv(src : ImageObj) → ImageObj`

Flip data vertically with `numpy.flipud()`

**Paramètres**

- src** – input image object

**Renvois**

Output image object

`class cdl.computation.image.GridParam`

Grid parameters

**direction**

Distribuer selon les. Sélection unique parmi : “col”, “row”. Par défaut : “col”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**cols**

Colonnes. Entier, non nul. Par défaut : 1.

**Type**

`guidata.dataset.dataitems.IntItem`

**rows**

Lignes. Entier, non nul. Par défaut : 1.

**Type**

`guidata.dataset.dataitems.IntItem`

**colspac**

Espace entre chaque colonne. Flottant supérieur à 0.0. Par défaut : 0.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**rowspac**

Espace entre chaque ligne. Flottant supérieur à 0.0. Par défaut : 0.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** **create**(*direction* : *str*, *cols* : *int*, *rows* : *int*, *colspac* : *float*, *rowspac* : *float*) →  
*cdl.computation.image.GridParam*

Renvoie une nouvelle instance de *GridParam* avec les champs initialisés aux valeurs données.

**Paramètres**

- **direction** (*str*) – Distribuer selon les. Sélection unique parmi : “col”, “row”. Par défaut : “col”.
- **cols** (*int*) – Colonnes. Entier, non nul. Par défaut : 1.
- **rows** (*int*) – Lignes. Entier, non nul. Par défaut : 1.
- **colspac** (*float*) – Espace entre chaque colonne. Flottant supérieur à 0.0. Par défaut : 0.0.
- **rowspac** (*float*) – Espace entre chaque ligne. Flottant supérieur à 0.0. Par défaut : 0.0.

**Renvoie**

Nouvelle instance de *GridParam*.

**class** `cdl.computation.image.ResizeParam`

Resize parameters

**zoom**

Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**mode**

Sélection unique parmi : “constant”, “nearest”, “reflect”, “wrap”. Par défaut : “constant”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**cval**

Valeur utilisée pour les points situés en dehors des frontières de l’image d’origine (si le mode est “constant”).  
Par défaut : 0.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**prefilter**

Par défaut : True.

**Type**

`guidata.dataset.dataitems.BoolItem`

**order**

Ordre. Ordre de l’interpolation de type spline. Entier compris entre 0 et 5. Par défaut : 3.

**Type**

`guidata.dataset.dataitems.IntItem`

**classmethod** `create`(*zoom* : *float*, *mode* : *str*, *cval* : *float*, *prefilter* : *bool*, *order* : *int*) → *cdl.computation.image.ResizeParam*

Renvoie une nouvelle instance de *ResizeParam* avec les champs initialisés aux valeurs données.

**Paramètres**

- **zoom** (*float*) – Par défaut : None.
- **mode** (*str*) – Sélection unique parmi : “constant”, “nearest”, “reflect”, “wrap”. Par défaut : “constant”.
- **cval** (*float*) – Valeur utilisée pour les points situés en dehors des frontières de l’image d’origine (si le mode est “constant”). Par défaut : 0.0.
- **prefilter** (*bool*) – Par défaut : True.
- **order** (*int*) – Ordre. Ordre de l’interpolation de type spline. Entier compris entre 0 et 5. Par défaut : 3.

**Renvoie**

Nouvelle instance de *ResizeParam*.

`cdl.computation.image.compute_resize`(*src* : *ImageObj*, *p* : *ResizeParam*) → *ImageObj*

Zooming function with `scipy.ndimage.zoom()`

**Paramètres**

- **src** – input image object
- **p** – parameters

**Renvoie**

Output image object

**class** `cdl.computation.image.BinningParam`

Binning parameters

**sx**

Nombre de pixels (X). Nombre de pixels adjacents à regrouper le long de l’axe des x. Entier supérieur à 2.  
Par défaut : 2.

**Type**

`guidata.dataset.dataitems.IntItem`

**sy**

Nombre de pixels (Y). Nombre de pixels adjacents à regrouper le long de l’axe des y. Entier supérieur à 2.  
Par défaut : 2.

**Type**

`guidata.dataset.dataitems.IntItem`

**operation**

Opération. Sélection unique parmi : “sum”, “average”, “median”, “min”, “max”. Par défaut : “sum”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**dtype\_str**

Type de données. Type de données de l’image générée. Sélection unique parmi : “dtype”, “float32”, “float64”, “complex128”, “int32”, “int16”, “uint16”, “uint8”. Par défaut : “dtype”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**change\_pixel\_size**

Modification de la taille des pixels de sorte que les dimensions de l’images restent les mêmes après l’opération. Par défaut : False.

**Type**

`guidata.dataset.dataitems.BoolItem`

**classmethod** `create`(*sx* : *int*, *sy* : *int*, *operation* : *str*, *dtype\_str* : *str*, *change\_pixel\_size* : *bool*) → *cdl.computation.image.BinningParam*

Renvoie une nouvelle instance de *BinningParam* avec les champs initialisés aux valeurs données.

**Paramètres**

- **sx** (*int*) – Nombre de pixels (X). Nombre de pixels adjacents à regrouper le long de l’axe des x. Entier supérieur à 2. Par défaut : 2.
- **sy** (*int*) – Nombre de pixels (Y). Nombre de pixels adjacents à regrouper le long de l’axe des y. Entier supérieur à 2. Par défaut : 2.
- **operation** (*str*) – Opération. Sélection unique parmi : “sum”, “average”, “median”, “min”, “max”. Par défaut : “sum”.
- **dtype\_str** (*str*) – Type de données. Type de données de l’image générée. Sélection unique parmi : “dtype”, “float32”, “float64”, “complex128”, “int32”, “int16”, “uint16”, “uint8”. Par défaut : “dtype”.
- **change\_pixel\_size** (*bool*) – Modification de la taille des pixels de sorte que les dimensions de l’images restent les mêmes après l’opération. Par défaut : False.

**Renvoie**

Nouvelle instance de *BinningParam*.

`cdl.computation.image.compute_binning`(*src* : *ImageObj*, *param* : *BinningParam*) → *ImageObj*

Binning function on data with *cdl.algorithms.image.binning()*

**Paramètres**

- **src** – input image object
- **param** – parameters

**Renvoie**

Output image object

`cdl.computation.image.extract_multiple_roi`(*src* : *ImageObj*, *group* : *DataSetGroup*) → *ImageObj*

Extract multiple regions of interest from data

**Paramètres**

- **src** – input image object
- **group** – parameters defining the regions of interest

**Renvoie**

Output image object

`cdl.computation.image.extract_single_roi`(*src* : *ImageObj*, *p* : *ROI2DParam*) → *ImageObj*

Extract single ROI

**Paramètres**

- **src** – input image object
- **p** – ROI parameters

**Renvoie**

Output image object

**class** `cdl.computation.image.LineProfileParam`

Horizontal or vertical profile parameters

**direction**

Sélection unique parmi : “horizontal”, “vertical”. Par défaut : “horizontal”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**row**

Ligne. Entier supérieur à 0. Par défaut : 0.

**Type**

`guidata.dataset.dataitems.IntItem`

**col**

Colonne. Entier supérieur à 0. Par défaut : 0.

**Type**

`guidata.dataset.dataitems.IntItem`

**classmethod** `create(direction : str, row : int, col : int) → cdl.computation.image.LineProfileParam`

Renvoie une nouvelle instance de `LineProfileParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **direction** (*str*) – Sélection unique parmi : “horizontal”, “vertical”. Par défaut : “horizontal”.
- **row** (*int*) – Ligne. Entier supérieur à 0. Par défaut : 0.
- **col** (*int*) – Colonne. Entier supérieur à 0. Par défaut : 0.

**Renvoie**

Nouvelle instance de `LineProfileParam`.

`cdl.computation.image.compute_line_profile(src : ImageObj, p : LineProfileParam) → SignalObj`

Compute horizontal or vertical profile

**Paramètres**

- **src** – input image object
- **p** – parameters

**Renvoie**

Signal object with the profile

**class** `cdl.computation.image.SegmentProfileParam`

Segment profile parameters

**row1**

Ligne (début). Entier supérieur à 0. Par défaut : 0.

**Type**

`guidata.dataset.dataitems.IntItem`

**col1**

Colonne (début). Entier supérieur à 0. Par défaut : 0.

**Type**

`guidata.dataset.dataitems.IntItem`

**row2**

Ligne (fin). Entier supérieur à 0. Par défaut : 0.

**Type**

`guidata.dataset.dataitems.IntItem`

**col2**

Colonne (fin). Entier supérieur à 0. Par défaut : 0.

**Type**

`guidata.dataset.dataitems.IntItem`

**classmethod** `create(row1 : int, col1 : int, row2 : int, col2 : int) →`  
`cdl.computation.image.SegmentProfileParam`

Renvoie une nouvelle instance de `SegmentProfileParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **row1** (*int*) – Ligne (début). Entier supérieur à 0. Par défaut : 0.
- **col1** (*int*) – Colonne (début). Entier supérieur à 0. Par défaut : 0.
- **row2** (*int*) – Ligne (fin). Entier supérieur à 0. Par défaut : 0.
- **col2** (*int*) – Colonne (fin). Entier supérieur à 0. Par défaut : 0.

**Renvoie**

Nouvelle instance de `SegmentProfileParam`.

`cdl.computation.image.compute_segment_profile(src : ImageObj, p : SegmentProfileParam) → SignalObj`

Compute segment profile

**Paramètres**

- **src** – input image object
- **p** – parameters

**Renvoie**

Signal object with the segment profile

**class** `cdl.computation.image.AverageProfileParam`

Average horizontal or vertical profile parameters

**direction**

Sélection unique parmi : “horizontal”, “vertical”. Par défaut : “horizontal”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**row1**

Ligne 1. Entier supérieur à 0. Par défaut : 0.

**Type**

`guidata.dataset.dataitems.IntItem`

**row2**

Ligne 2. Entier supérieur à -1. Par défaut : -1.

**Type**

`guidata.dataset.dataitems.IntItem`

**col1**

Colonne 1. Entier supérieur à 0. Par défaut : 0.

**Type**

`guidata.dataset.dataitems.IntItem`

**col2**

Colonne 2. Entier supérieur à -1. Par défaut : -1.

**Type**

`guidata.dataset.dataitems.IntItem`

**classmethod create**(*direction* : *str*, *row1* : *int*, *row2* : *int*, *col1* : *int*, *col2* : *int*) →

`cdl.computation.image.AverageProfileParam`

Renvoie une nouvelle instance de `AverageProfileParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **direction** (*str*) – Sélection unique parmi : “horizontal”, “vertical”. Par défaut : “horizontal”.
- **row1** (*int*) – Ligne 1. Entier supérieur à 0. Par défaut : 0.
- **row2** (*int*) – Ligne 2. Entier supérieur à -1. Par défaut : -1.
- **col1** (*int*) – Colonne 1. Entier supérieur à 0. Par défaut : 0.
- **col2** (*int*) – Colonne 2. Entier supérieur à -1. Par défaut : -1.

**Renvoie**

Nouvelle instance de `AverageProfileParam`.

`cdl.computation.image.compute_average_profile`(*src* : `ImageObj`, *p* : `AverageProfileParam`) → `SignalObj`

Compute horizontal or vertical average profile

**Paramètres**

- **src** – input image object
- **p** – parameters

**Renvoie**

Signal object with the average profile

**class** `cdl.computation.image.RadialProfileParam`

Radial profile parameters

**center**

Position du centre. Sélection unique parmi : “centroid”, “center”, “user”. Par défaut : “centroid”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**x0**

$X_{\text{Centre}}$ . Flottant, unité : pixel. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**y0**

$X_{\text{Centre}}$ . Flottant, unité : pixel. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod create**(*center* : *str*, *x0* : *float*, *y0* : *float*) → `cdl.computation.image.RadialProfileParam`

Renvoie une nouvelle instance de `RadialProfileParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **center** (*str*) – Position du centre. Sélection unique parmi : “centroid”, “center”, “user”. Par défaut : “centroid”.
- **x0** (*float*) –  $X_{\text{Centre}}$ . Flottant, unité : pixel. Par défaut : None.
- **y0** (*float*) –  $X_{\text{Centre}}$ . Flottant, unité : pixel. Par défaut : None.

**Renvoie**

Nouvelle instance de `RadialProfileParam`.

**update\_from\_image**(*obj* : ImageObj) → None

Update parameters from image

**choice\_callback**(*item*, *value*)

Callback for choice item

**cdl.computation.image.compute\_radial\_profile**(*src* : ImageObj, *p* : RadialProfileParam) → *SignalObj*

Compute radial profile around the centroid with `cdl.algorithms.image.get_radial_profile()`

**Paramètres**

— **src** – input image object

— **p** – parameters

**Renvoie**

Signal object with the radial profile

**cdl.computation.image.compute\_histogram**(*src* : ImageObj, *p* : HistogramParam) → *SignalObj*

Compute histogram of the image data, with `numpy.histogram()`

**Paramètres**

— **src** – input image object

— **p** – parameters

**Renvoie**

Signal object with the histogram

**cdl.computation.image.compute\_swap\_axes**(*src* : ImageObj) → *ImageObj*

Swap image axes with `numpy.transpose()`

**Paramètres**

**src** – input image object

**Renvoie**

Output image object

**cdl.computation.image.compute\_abs**(*src* : ImageObj) → *ImageObj*

Compute absolute value with `numpy.absolute`

**Paramètres**

**src** – input image object

**Renvoie**

Output image object

**cdl.computation.image.compute\_re**(*src* : ImageObj) → *ImageObj*

Compute real part with `numpy.real()`

**Paramètres**

**src** – input image object

**Renvoie**

Output image object

**cdl.computation.image.compute\_im**(*src* : ImageObj) → *ImageObj*

Compute imaginary part with `numpy.imag()`

**Paramètres**

**src** – input image object

**Renvoie**

Output image object

**class** **cdl.computation.image.DataTypeIParam**

Convert image data type parameters

**dtype\_str**

Type de données de destination. Type de données de l'image générée. Sélection unique parmi : "float32", "float64", "complex128", "int32", "int16", "uint16", "uint8". Par défaut : "float32".

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**classmethod** `create(dtype_str : str) → cdl.computation.image.DataTypeIParam`

Renvoie une nouvelle instance de `DataTypeIParam` avec les champs initialisés aux valeurs données.

**Paramètres**

**dtype\_str** (*str*) – Type de données de destination. Type de données de l'image générée. Sélection unique parmi : "float32", "float64", "complex128", "int32", "int16", "uint16", "uint8". Par défaut : "float32".

**Renvoie**

Nouvelle instance de `DataTypeIParam`.

`cdl.computation.image.compute_astype(src : ImageObj, p : DataTypeIParam) → ImageObj`

Convert image data type with `cdl.algorithms.datatypes.clip_astype()`

**Paramètres**

- **src** – input image object
- **p** – parameters

**Renvoie**

Output image object

`cdl.computation.image.compute_log10(src : ImageObj) → ImageObj`

Compute log10 with `numpy.log10`

**Paramètres**

**src** – input image object

**Renvoie**

Output image object

`cdl.computation.image.compute_exp(src : ImageObj) → ImageObj`

Compute exponential with `numpy.exp`

**Paramètres**

**src** – input image object

**Renvoie**

Output image object

**class** `cdl.computation.image.ZCalibrateParam`

Image linear calibration parameters

**a**

Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**b**

Par défaut : 0.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(a : float, b : float) → cdl.computation.image.ZCalibrateParam`

Renvoie une nouvelle instance de `ZCalibrateParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **a** (*float*) – Par défaut : 1.0.
- **b** (*float*) – Par défaut : 0.0.

**Renvoie**

Nouvelle instance de *ZCalibrateParam*.

`cdl.computation.image.compute_calibration(src : ImageObj, p : ZCalibrateParam) → ImageObj`

Compute linear calibration

**Paramètres**

- **src** – input image object
- **p** – calibration parameters

**Renvoie**

Output image object

`cdl.computation.image.compute_clip(src : ImageObj, p : ClipParam) → ImageObj`

Apply clipping with `numpy.clip()`

**Paramètres**

- **src** – input image object
- **p** – parameters

**Renvoie**

Output image object

`cdl.computation.image.compute_offset_correction(src : ImageObj, p : ROI2DParam) → ImageObj`

Apply offset correction

**Paramètres**

- **src** – input image object
- **p** – parameters

**Renvoie**

Output image object

`cdl.computation.image.compute_gaussian_filter(src : ImageObj, p : GaussianParam) → ImageObj`

Compute gaussian filter with `scipy.ndimage.gaussian_filter()`

**Paramètres**

- **src** – input image object
- **p** – parameters

**Renvoie**

Output image object

`cdl.computation.image.compute_moving_average(src : ImageObj, p : MovingAverageParam) → ImageObj`

Compute moving average with `scipy.ndimage.uniform_filter()`

**Paramètres**

- **src** – input image object
- **p** – parameters

**Renvoie**

Output image object

`cdl.computation.image.compute_moving_median(src : ImageObj, p : MovingMedianParam) → ImageObj`

Compute moving median with `scipy.ndimage.median_filter()`

**Paramètres**

- **src** – input image object
- **p** – parameters

**Renvoie**

Output image object

`cdl.computation.image.compute_wiener(src : ImageObj) → ImageObj`

Compute Wiener filter with `scipy.signal.wiener()`

**Paramètres**

— **src** – input image object

**Renvoie**

Output image object

`cdl.computation.image.compute_fft(src : ImageObj, p : FFTParam | None = None) → ImageObj`

Compute FFT with `cdl.algorithms.image.fft2d()`

**Paramètres**

— **src** – input image object

— **p** – parameters

**Renvoie**

Output image object

`cdl.computation.image.compute_ifft(src : ImageObj, p : FFTParam | None = None) → ImageObj`

Compute inverse FFT with `cdl.algorithms.image.ifft2d()`

**Paramètres**

— **src** – input image object

— **p** – parameters

**Renvoie**

Output image object

`cdl.computation.image.compute_magnitude_spectrum(src : ImageObj, p : SpectrumParam | None = None) → ImageObj`

Compute magnitude spectrum with `cdl.algorithms.image.magnitude_spectrum()`

**Paramètres**

— **src** – input image object

— **p** – parameters

**Renvoie**

Output image object

`cdl.computation.image.compute_phase_spectrum(src : ImageObj) → ImageObj`

Compute phase spectrum with `cdl.algorithms.image.phase_spectrum()`

**Paramètres**

— **src** – input image object

**Renvoie**

Output image object

`cdl.computation.image.compute_psd(src : ImageObj, p : SpectrumParam | None = None) → ImageObj`

Compute power spectral density with `cdl.algorithms.image.psd()`

**Paramètres**

— **src** – input image object

— **p** – parameters

**Renvoie**

Output image object

`class cdl.computation.image.ButterworthParam`

Butterworth filter parameters

**cut\_off**

Fréquence de coupure relative. Fréquence de coupure relative. Flottant compris entre 0.0 et 0.5. Par défaut : 0.005.

**Type**`guidata.dataset.dataitems.FloatItem`**high\_pass**

Si vrai, appliquer un filtre passe-haut au lieu d'un filtre passe-bas. Par défaut : False.

**Type**`guidata.dataset.dataitems.BoolItem`**order**

Ordre. Ordre du filtre de butterworth. Entier supérieur à 1. Par défaut : 2.

**Type**`guidata.dataset.dataitems.IntItem`

**classmethod** **create**(*cut\_off* : *float*, *high\_pass* : *bool*, *order* : *int*) →  
`cdl.computation.image.ButterworthParam`

Renvoie une nouvelle instance de `ButterworthParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **cut\_off** (*float*) – Fréquence de coupure relative. Fréquence de coupure relative. Flottant compris entre 0.0 et 0.5. Par défaut : 0.005.
- **high\_pass** (*bool*) – Si vrai, appliquer un filtre passe-haut au lieu d'un filtre passe-bas. Par défaut : False.
- **order** (*int*) – Ordre. Ordre du filtre de butterworth. Entier supérieur à 1. Par défaut : 2.

**Renvoie**

Nouvelle instance de `ButterworthParam`.

`cdl.computation.image.compute_butterworth`(*src* : `ImageObj`, *p* : `ButterworthParam`) → `ImageObj`

Compute Butterworth filter with `skimage.filters.butterworth()`

**Paramètres**

- **src** – input image object
- **p** – parameters

**Renvoie**

Output image object

`cdl.computation.image.calc_resultshape`(*title* : *str*, *shape* : `Literal`['rectangle', 'circle', 'ellipse', 'segment', 'marker', 'point', 'polygon'], *obj* : `ImageObj`, *func* : `Callable`, *\*args* : *Any*, *add\_label* : *bool* = *False*) → `ResultShape` | *None*

Calculate result shape by executing a computation function on an image object, taking into account the image origin (x0, y0), scale (dx, dy) and ROIs.

**Paramètres**

- **title** – result title
- **shape** – result shape kind
- **obj** – input image object
- **func** – computation function
- **\*args** – computation function arguments
- **add\_label** – if True, add a label item (and the geometrical shape) to plot (default to False)

**Renvoie**

Result shape object or None if no result is found

**Avertissement :** The computation function must take either a single argument (the data) or multiple arguments (the data followed by the computation parameters).

Moreover, the computation function must return a single value or a NumPy array containing the result of the computation. This array contains the coordinates of points, polygons, circles or ellipses in the form `[[x, y],`

..., or `[[x0, y0, x1, y1, ...], ...]`, or `[[x0, y0, r], ...]`, or `[[x0, y0, a, b, theta], ...]`.

`cdl.computation.image.get_centroid_coords(data : ndarray) → ndarray`

Return centroid coordinates with `cdl.algorithms.image.get_centroid_fourier()`

**Paramètres**

**data** – input data

**Renvoie**

Centroid coordinates

`cdl.computation.image.compute_centroid(image : ImageObj) → ResultShape | None`

Compute centroid with `cdl.algorithms.image.get_centroid_fourier()`

**Paramètres**

**image** – input image

**Renvoie**

Centroid coordinates

`cdl.computation.image.get_enclosing_circle_coords(data : ndarray) → ndarray`

Return diameter coords for the circle contour enclosing image values above threshold (FWHM)

**Paramètres**

**data** – input data

**Renvoie**

Diameter coords

`cdl.computation.image.compute_enclosing_circle(image : ImageObj) → ResultShape | None`

Compute minimum enclosing circle with `cdl.algorithms.image.get_enclosing_circle()`

**Paramètres**

**image** – input image

**Renvoie**

Diameter coords

**class** `cdl.computation.image.HoughCircleParam`

Circle Hough transform parameters

**min\_radius**

Rayon<sub>min</sub>. Entier supérieur à 0, non nul, unité : pixels. Par défaut : None.

**Type**

`guidata.dataset.dataitems.IntItem`

**max\_radius**

Rayon<sub>max</sub>. Entier supérieur à 0, non nul, unité : pixels. Par défaut : None.

**Type**

`guidata.dataset.dataitems.IntItem`

**min\_distance**

Distance minimale. Entier supérieur à 0. Par défaut : None.

**Type**

`guidata.dataset.dataitems.IntItem`

**classmethod** `create(min_radius : int, max_radius : int, min_distance : int) →`

`cdl.computation.image.HoughCircleParam`

Renvoie une nouvelle instance de `HoughCircleParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **min\_radius** (*int*) – Rayon<sub>min</sub>. Entier supérieur à 0, non nul, unité : pixels. Par défaut : None.
- **max\_radius** (*int*) – Rayon<sub>max</sub>. Entier supérieur à 0, non nul, unité : pixels. Par défaut : None.
- **min\_distance** (*int*) – Distance minimale. Entier supérieur à 0. Par défaut : None.

**Renvoie**

Nouvelle instance de *HoughCircleParam*.

`cdl.computation.image.compute_hough_circle_peaks(image : ImageObj, p : HoughCircleParam) → ResultShape | None`

Compute Hough circles with *cdl.algorithms.image.get\_hough\_circle\_peaks()*

**Paramètres**

- **image** – input image
- **p** – parameters

**Renvoie**

Circle coordinates

`cdl.computation.image.compute_stats(obj : ImageObj) → ResultProperties`

Compute statistics on an image

**Paramètres**

- obj** – input image object

**Renvoie**

Result properties

**Threshold features****Threshold computation module**

**class** `cdl.computation.image.threshold.ThresholdParam`

Histogram threshold parameters

**method**

Méthode de seuillage. Sélection unique parmi : “manual”, “isodata”, “li”, “mean”, “minimum”, “otsu”, “triangle”, “yen”. Par défaut : “manual”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**bins**

Nombre de classes. Entier supérieur à 1. Par défaut : 256.

**Type**

`guidata.dataset.dataitems.IntItem`

**value**

Valeur de seuil. Par défaut : 0.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**operation**

Opération. Sélection unique parmi : “>”, “<”. Par défaut : “>”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**classmethod** `create(method : str, bins : int, value : float, operation : str) →`  
`cdl.computation.image.threshold.ThresholdParam`

Renvoie une nouvelle instance de `ThresholdParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **method** (`str`) – Méthode de seuillage. Sélection unique parmi : “manual”, “isodata”, “li”, “mean”, “minimum”, “otsu”, “triangle”, “yen”. Par défaut : “manual”.
- **bins** (`int`) – Nombre de classes. Entier supérieur à 1. Par défaut : 256.
- **value** (`float`) – Valeur de seuil. Par défaut : 0.0.
- **operation** (`str`) – Opération. Sélection unique parmi : “>”, “<”. Par défaut : “>”.

**Renvoie**

Nouvelle instance de `ThresholdParam`.

`cdl.computation.image.threshold.compute_threshold(src : ImageObj, p : ThresholdParam) → ImageObj`

Compute the threshold, using one of the available algorithms :

- Manual : a fixed threshold value
- ISODATA : `skimage.filters.threshold_isodata()`
- Li : `skimage.filters.threshold_li()`
- Mean : `skimage.filters.threshold_mean()`
- Minimum : `skimage.filters.threshold_minimum()`
- Otsu : `skimage.filters.threshold_otsu()`
- Triangle : `skimage.filters.threshold_triangle()`
- Yen : `skimage.filters.threshold_yen()`

**Paramètres**

- **src** – input image object
- **p** – parameters

**Renvoie**

Output image object

`cdl.computation.image.threshold.compute_threshold_isodata(src : ImageObj) → ImageObj`

Compute the threshold using the Isodata algorithm with default parameters, see `skimage.filters.threshold_isodata()`

**Paramètres**

- src** – input image object

**Renvoie**

Output image object

`cdl.computation.image.threshold.compute_threshold_li(src : ImageObj) → ImageObj`

Compute the threshold using the Li algorithm with default parameters, see `skimage.filters.threshold_li()`

**Paramètres**

- src** – input image object

**Renvoie**

Output image object

`cdl.computation.image.threshold.compute_threshold_mean(src : ImageObj) → ImageObj`

Compute the threshold using the Mean algorithm, see `skimage.filters.threshold_mean()`

**Paramètres**

- src** – input image object

**Renvoie**

Output image object

`cdl.computation.image.threshold.compute_threshold_minimum(src : ImageObj) → ImageObj`

Compute the threshold using the Minimum algorithm with default parameters, see `skimage.filters.threshold_minimum()`

**Paramètres**

**src** – input image object

**Renvoie**

Output image object

`cdl.computation.image.threshold.compute_threshold_otsu(src : ImageObj) → ImageObj`

Compute the threshold using the Otsu algorithm with default parameters, see `skimage.filters.threshold_otsu()`

**Paramètres**

**src** – input image object

**Renvoie**

Output image object

`cdl.computation.image.threshold.compute_threshold_triangle(src : ImageObj) → ImageObj`

Compute the threshold using the Triangle algorithm with default parameters, see `skimage.filters.threshold_triangle()`

**Paramètres**

**src** – input image object

**Renvoie**

Output image object

`cdl.computation.image.threshold.compute_threshold_yen(src : ImageObj) → ImageObj`

Compute the threshold using the Yen algorithm with default parameters, see `skimage.filters.threshold_yen()`

**Paramètres**

**src** – input image object

**Renvoie**

Output image object

## Exposure correction features

### Exposure computation module

**class** `cdl.computation.image.exposure.AdjustGammaParam`

Gamma adjustment parameters

**gamma**

Facteur de correction gamma (plus la valeur est élevée, plus le contraste est important). Flottant supérieur à 0.0. Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**gain**

Facteur de gain (plus la valeur est élevée, plus le contraste est important). Flottant supérieur à 0.0. Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(gamma : float, gain : float) → cdl.computation.image.exposure.AdjustGammaParam`

Renvoie une nouvelle instance de `AdjustGammaParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **gamma** (*float*) – Facteur de correction gamma (plus la valeur est élevée, plus le contraste est important). Flottant supérieur à 0.0. Par défaut : 1.0.
- **gain** (*float*) – Facteur de gain (plus la valeur est élevée, plus le contraste est important). Flottant supérieur à 0.0. Par défaut : 1.0.

**Renvoie**

Nouvelle instance de `AdjustGammaParam`.

`cdl.computation.image.exposure.compute_adjust_gamma(src : ImageObj, p : AdjustGammaParam) → ImageObj`

Gamma correction with `skimage.exposure.adjust_gamma()`

**Paramètres**

- **src** – input image object
- **p** – parameters

**Renvoie**

Output image object

**class** `cdl.computation.image.exposure.AdjustLogParam`

Logarithmic adjustment parameters

**gain**

Facteur de gain (plus la valeur est élevée, plus le contraste est important). Flottant supérieur à 0.0. Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**inv**

Si vrai, appliquer une transformation logarithmique inverse. Par défaut : False.

**Type**

`guidata.dataset.dataitems.BoolItem`

**classmethod** `create(gain : float, inv : bool) → cdl.computation.image.exposure.AdjustLogParam`

Renvoie une nouvelle instance de `AdjustLogParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **gain** (*float*) – Facteur de gain (plus la valeur est élevée, plus le contraste est important). Flottant supérieur à 0.0. Par défaut : 1.0.
- **inv** (*bool*) – Si vrai, appliquer une transformation logarithmique inverse. Par défaut : False.

**Renvoie**

Nouvelle instance de `AdjustLogParam`.

`cdl.computation.image.exposure.compute_adjust_log(src : ImageObj, p : AdjustLogParam) → ImageObj`

Compute log correction with `skimage.exposure.adjust_log()`

**Paramètres**

- **src** – input image object
- **p** – parameters

**Renvoie**

Output image object

**class** `cdl.computation.image.exposure.AdjustSigmoidParam`

Sigmoid adjustment parameters

**cutoff**

Valeur de coupure (plus la valeur est élevée, plus le contraste est important). Flottant compris entre 0.0 et 1.0. Par défaut : 0.5.

**Type**

`guidata.dataset.dataitems.FloatItem`

**gain**

Facteur de gain (plus la valeur est élevée, plus le contraste est important). Flottant supérieur à 0.0. Par défaut : 10.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**inv**

Si vrai, appliquer une transformation sigmoïde inverse. Par défaut : False.

**Type**

`guidata.dataset.dataitems.BoolItem`

**classmethod** `create(cutoff : float, gain : float, inv : bool) →`

`cdl.computation.image.exposure.AdjustSigmoidParam`

Renvoie une nouvelle instance de `AdjustSigmoidParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **cutoff** (*float*) – Valeur de coupure (plus la valeur est élevée, plus le contraste est important). Flottant compris entre 0.0 et 1.0. Par défaut : 0.5.
- **gain** (*float*) – Facteur de gain (plus la valeur est élevée, plus le contraste est important). Flottant supérieur à 0.0. Par défaut : 10.0.
- **inv** (*bool*) – Si vrai, appliquer une transformation sigmoïde inverse. Par défaut : False.

**Renvoie**

Nouvelle instance de `AdjustSigmoidParam`.

`cdl.computation.image.exposure.compute_adjust_sigmoid(src : ImageObj, p : AdjustSigmoidParam) →`  
`ImageObj`

Compute sigmoid correction with `skimage.exposure.adjust_sigmoid()`

**Paramètres**

- **src** – input image object
- **p** – parameters

**Renvoie**

Output image object

**class** `cdl.computation.image.exposure.RescaleIntensityParam`

Intensity rescaling parameters

**in\_range**

Echelle de niveaux en entrée. Valeurs min/max d'intensité de l'image en entrée ("image" correspond aux niveaux min/max de l'image en entrée, "dtype" correspond aux valeurs min/max du type de données de l'image). Sélection unique parmi : "image", "dtype", "float32", "float64", "complex128", "int32", "int16", "uint16", "uint8". Par défaut : "image".

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**out\_range**

Echelle de niveaux en sortie. Valeurs min/max d'intensité de l'image en sortie ("image" correspond aux niveaux min/max de l'image en entrée, "dtype" correspond aux valeurs min/max du type de données de l'image). Sélection unique parmi : "image", "dtype", "float32", "float64", "complex128", "int32", "int16", "uint16", "uint8". Par défaut : "dtype".

**Type**`guidata.dataset.dataitems.ChoiceItem`

**classmethod** `create(in_range : str, out_range : str) →`  
`cdl.computation.image.exposure.RescaleIntensityParam`

Renvoie une nouvelle instance de `RescaleIntensityParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **in\_range** (`str`) – Echelle de niveaux en entrée. Valeurs min/max d'intensité de l'image en entrée ("image" correspond aux niveaux min/max de l'image en entrée, "dtype" correspond aux valeurs min/max du type de données de l'image). Sélection unique parmi : "image", "dtype", "float32", "float64", "complex128", "int32", "int16", "uint16", "uint8". Par défaut : "image".
- **out\_range** (`str`) – Echelle de niveaux en sortie. Valeurs min/max d'intensité de l'image en sortie ("image" correspond aux niveaux min/max de l'image en entrée, "dtype" correspond aux valeurs min/max du type de données de l'image). Sélection unique parmi : "image", "dtype", "float32", "float64", "complex128", "int32", "int16", "uint16", "uint8". Par défaut : "dtype".

**Renvoie**

Nouvelle instance de `RescaleIntensityParam`.

`cdl.computation.image.exposure.compute_rescale_intensity(src : ImageObj, p :`  
`RescaleIntensityParam) → ImageObj`

Rescale image intensity levels with `skimage.exposure.rescale_intensity()`

**Paramètres**

- **src** – input image object
- **p** – parameters

**Renvoie**

Output image object

**class** `cdl.computation.image.exposure.EqualizeHistParam`

Histogram equalization parameters

**nbins**

Nombre de classes. Nombre de classes de l'histogramme des niveaux de l'image. Entier supérieur à 1. Par défaut : 256.

**Type**`guidata.dataset.dataitems.IntItem`

**classmethod** `create(nbins : int) → cdl.computation.image.exposure.EqualizeHistParam`

Renvoie une nouvelle instance de `EqualizeHistParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- nbins** (`int`) – Nombre de classes. Nombre de classes de l'histogramme des niveaux de l'image. Entier supérieur à 1. Par défaut : 256.

**Renvoie**

Nouvelle instance de `EqualizeHistParam`.

`cdl.computation.image.exposure.compute_equalize_hist(src : ImageObj, p : EqualizeHistParam) →`  
`ImageObj`

Histogram equalization with `skimage.exposure.equalize_hist()`

**Paramètres**

- **src** – input image object
- **p** – parameters

**Renvoie**

Output image object

**class** `cdl.computation.image.exposure.EqualizeAdaptHistParam`

Adaptive histogram equalization parameters

**nbins**

Nombre de classes. Nombre de classes de l'histogramme des niveaux de l'image. Entier supérieur à 1. Par défaut : 256.

**Type**

`guidata.dataset.dataitems.IntItem`

**clip\_limit**

Écrêtage limite. Valeur d'écrtage (plus la valeur est élevée, plus le contraste est important). Flottant compris entre 0.0 et 1.0. Par défaut : 0.01.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(nbins : int, clip_limit : float) →`

`cdl.computation.image.exposure.EqualizeAdaptHistParam`

Renvoie une nouvelle instance de `EqualizeAdaptHistParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **nbins** (*int*) – Nombre de classes. Nombre de classes de l'histogramme des niveaux de l'image. Entier supérieur à 1. Par défaut : 256.
- **clip\_limit** (*float*) – Écrêtage limite. Valeur d'écrtage (plus la valeur est élevée, plus le contraste est important). Flottant compris entre 0.0 et 1.0. Par défaut : 0.01.

**Renvoie**

Nouvelle instance de `EqualizeAdaptHistParam`.

`cdl.computation.image.exposure.compute_equalize_adapthist(src : ImageObj, p :`

`EqualizeAdaptHistParam) → ImageObj`

Adaptive histogram equalization with `skimage.exposure.equalize_adapthist()`

**Paramètres**

- **src** – input image object
- **p** – parameters

**Renvoie**

Output image object

## Restoration features

### Restoration computation module

**class** `cdl.computation.image.restoration.DenoiseTVParam`

Total Variation denoising parameters

**weight**

Poids de débruitage. Plus le poids est élevé, plus le débruitage est fort (aux dépens de la fidélité des données). Flottant supérieur à 0, non nul. Par défaut : 0.1.

**Type**

`guidata.dataset.dataitems.FloatItem`

**eps**

Epsilon. Différence relative de la valeur de la fonction de coût qui détermine le critère d'arrêt de l'algorithme. Ce dernier s'arrête lorsque :  $(e_{(n-1)} - e_n) < \text{eps} * e_0$ . Flottant supérieur à 0, non nul. Par défaut : 0.0002.

**Type**

`guidata.dataset.dataitems.FloatItem`

**max\_num\_iter**

Nb. max. d'itérations. Nombre maximal d'itérations utilisé pour l'optimisation. Entier supérieur à 0, non nul. Par défaut : 200.

**Type**

`guidata.dataset.dataitems.IntItem`

**classmethod** `create(weight : float, eps : float, max_num_iter : int) →`

`cdl.computation.image.restoration.DenoiseTVParam`

Renvoie une nouvelle instance de `DenoiseTVParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **weight** (*float*) – Poids de débruitage. Plus le poids est élevé, plus le débruitage est fort (aux dépens de la fidélité des données). Flottant supérieur à 0, non nul. Par défaut : 0.1.
- **eps** (*float*) – Epsilon. Différence relative de la valeur de la fonction de coût qui détermine le critère d'arrêt de l'algorithme. Ce dernier s'arrête lorsque :  $(e_{(n-1)} - e_n) < \text{eps} * e_0$ . Flottant supérieur à 0, non nul. Par défaut : 0.0002.
- **max\_num\_iter** (*int*) – Nb. max. d'itérations. Nombre maximal d'itérations utilisé pour l'optimisation. Entier supérieur à 0, non nul. Par défaut : 200.

**Renvoie**

Nouvelle instance de `DenoiseTVParam`.

`cdl.computation.image.restoration.compute_denoise_tv(src : ImageObj, p : DenoiseTVParam) → ImageObj`

Compute Total Variation denoising with `skimage.restoration.denoise_tv_chambolle()`

**Paramètres**

- **src** – input image object
- **p** – parameters

**Renvoie**

Output image object

**class** `cdl.computation.image.restoration.DenoiseBilateralParam`

Bilateral filter denoising parameters

**sigma\_spatial**

<sup>spatial</sup>. Ecart-type dans le domaine spatial. Une valeur élevée a pour effet de moyenner des pixels séparés par de grandes distances. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**mode**

Sélection unique parmi : “constant”, “edge”, “symmetric”, “reflect”, “wrap”. Par défaut : “constant”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**cval**

Valeur de remplissage en dehors des bornes de l'image (en mode constant). Par défaut : 0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(sigma_spatial : float, mode : str, cval : float) →`  
`cdl.computation.image.restoration.DenoiseBilateralParam`

Renvoie une nouvelle instance de `DenoiseBilateralParam` avec les champs initialisés aux valeurs données.

#### Paramètres

- **sigma\_spatial** (*float*) – <sup>spatial</sup>. Ecart-type dans le domaine spatial. Une valeur élevée a pour effet de moyenner des pixels séparés par de grandes distances. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 1.0.
- **mode** (*str*) – Sélection unique parmi : “constant”, “edge”, “symmetric”, “reflect”, “wrap”. Par défaut : “constant”.
- **cval** (*float*) – Valeur de remplissage en dehors des bornes de l’image (en mode constant). Par défaut : 0.

#### Renvoie

Nouvelle instance de `DenoiseBilateralParam`.

`cdl.computation.image.restoration.compute_denoise_bilateral(src : ImageObj, p :`  
`DenoiseBilateralParam) → ImageObj`

Compute bilateral filter denoising with `skimage.restoration.denoise_bilateral()`

#### Paramètres

- **src** – input image object
- **p** – parameters

#### Renvoie

Output image object

**class** `cdl.computation.image.restoration.DenoiseWaveletParam`

Wavelet denoising parameters

#### wavelet

Ondelette. Sélection unique parmi : “bior1.1”, “bior1.3”, “bior1.5”, “bior2.2”, “bior2.4”, “bior2.6”, “bior2.8”, “bior3.1”, “bior3.3”, “bior3.5”, “bior3.7”, “bior3.9”, “bior4.4”, “bior5.5”, “bior6.8”, “cgau1”, “cgau2”, “cgau3”, “cgau4”, “cgau5”, “cgau6”, “cgau7”, “cgau8”, “cmor”, “coif1”, “coif2”, “coif3”, “coif4”, “coif5”, “coif6”, “coif7”, “coif8”, “coif9”, “coif10”, “coif11”, “coif12”, “coif13”, “coif14”, “coif15”, “coif16”, “coif17”, “db1”, “db2”, “db3”, “db4”, “db5”, “db6”, “db7”, “db8”, “db9”, “db10”, “db11”, “db12”, “db13”, “db14”, “db15”, “db16”, “db17”, “db18”, “db19”, “db20”, “db21”, “db22”, “db23”, “db24”, “db25”, “db26”, “db27”, “db28”, “db29”, “db30”, “db31”, “db32”, “db33”, “db34”, “db35”, “db36”, “db37”, “db38”, “dmey”, “fbsp”, “gaus1”, “gaus2”, “gaus3”, “gaus4”, “gaus5”, “gaus6”, “gaus7”, “gaus8”, “haar”, “mexh”, “morl”, “rbio1.1”, “rbio1.3”, “rbio1.5”, “rbio2.2”, “rbio2.4”, “rbio2.6”, “rbio2.8”, “rbio3.1”, “rbio3.3”, “rbio3.5”, “rbio3.7”, “rbio3.9”, “rbio4.4”, “rbio5.5”, “rbio6.8”, “shan”, “sym2”, “sym3”, “sym4”, “sym5”, “sym6”, “sym7”, “sym8”, “sym9”, “sym10”, “sym11”, “sym12”, “sym13”, “sym14”, “sym15”, “sym16”, “sym17”, “sym18”, “sym19”, “sym20”. Par défaut : “sym9”.

#### Type

`guidata.dataset.dataitems.ChoiceItem`

#### mode

Sélection unique parmi : “soft”, “hard”. Par défaut : “soft”.

#### Type

`guidata.dataset.dataitems.ChoiceItem`

#### method

Méthode. Sélection unique parmi : “BayesShrink”, “VisuShrink”. Par défaut : “VisuShrink”.

#### Type

`guidata.dataset.dataitems.ChoiceItem`

**classmethod create**(*wavelet* : *str*, *mode* : *str*, *method* : *str*) →  
*cdl.computation.image.restoration.DenoiseWaveletParam*

Renvoie une nouvelle instance de *DenoiseWaveletParam* avec les champs initialisés aux valeurs données.

#### Paramètres

- **wavelet** (*str*) – Ondelette. Sélection unique parmi : “bior1.1”, “bior1.3”, “bior1.5”, “bior2.2”, “bior2.4”, “bior2.6”, “bior2.8”, “bior3.1”, “bior3.3”, “bior3.5”, “bior3.7”, “bior3.9”, “bior4.4”, “bior5.5”, “bior6.8”, “cgau1”, “cgau2”, “cgau3”, “cgau4”, “cgau5”, “cgau6”, “cgau7”, “cgau8”, “cmor”, “coif1”, “coif2”, “coif3”, “coif4”, “coif5”, “coif6”, “coif7”, “coif8”, “coif9”, “coif10”, “coif11”, “coif12”, “coif13”, “coif14”, “coif15”, “coif16”, “coif17”, “db1”, “db2”, “db3”, “db4”, “db5”, “db6”, “db7”, “db8”, “db9”, “db10”, “db11”, “db12”, “db13”, “db14”, “db15”, “db16”, “db17”, “db18”, “db19”, “db20”, “db21”, “db22”, “db23”, “db24”, “db25”, “db26”, “db27”, “db28”, “db29”, “db30”, “db31”, “db32”, “db33”, “db34”, “db35”, “db36”, “db37”, “db38”, “dmey”, “fbsp”, “gaus1”, “gaus2”, “gaus3”, “gaus4”, “gaus5”, “gaus6”, “gaus7”, “gaus8”, “haar”, “mexh”, “morl”, “rbio1.1”, “rbio1.3”, “rbio1.5”, “rbio2.2”, “rbio2.4”, “rbio2.6”, “rbio2.8”, “rbio3.1”, “rbio3.3”, “rbio3.5”, “rbio3.7”, “rbio3.9”, “rbio4.4”, “rbio5.5”, “rbio6.8”, “shan”, “sym2”, “sym3”, “sym4”, “sym5”, “sym6”, “sym7”, “sym8”, “sym9”, “sym10”, “sym11”, “sym12”, “sym13”, “sym14”, “sym15”, “sym16”, “sym17”, “sym18”, “sym19”, “sym20”. Par défaut : “sym9”.
- **mode** (*str*) – Sélection unique parmi : “soft”, “hard”. Par défaut : “soft”.
- **method** (*str*) – Méthode. Sélection unique parmi : “BayesShrink”, “VisuShrink”. Par défaut : “VisuShrink”.

#### Renvoie

Nouvelle instance de *DenoiseWaveletParam*.

*cdl.computation.image.restoration.compute\_denoise\_wavelet*(*src* : *ImageObj*, *p* :  
*DenoiseWaveletParam*) → *ImageObj*

Compute Wavelet denoising with *skimage.restoration.denoise\_wavelet*()

#### Paramètres

- **src** – input image object
- **p** – parameters

#### Renvoie

Output image object

*cdl.computation.image.restoration.compute\_denoise\_tophat*(*src* : *ImageObj*, *p* : *MorphologyParam*)  
→ *ImageObj*

Denoise using White Top-Hat with *skimage.morphology.white\_tophat*()

#### Paramètres

- **src** – input image object
- **p** – parameters

#### Renvoie

Output image object

## Morphological features

### Morphology computation module

**class** `cdl.computation.image.morphology.MorphologyParam`

White Top-Hat parameters

**radius**

Rayon. Rayon de l'ouverture circulaire (disque). Entier supérieur à 1. Par défaut : 1.

**Type**

`guidata.dataset.dataitems.IntItem`

**classmethod** `create(radius : int) → cdl.computation.image.morphology.MorphologyParam`

Renvoie une nouvelle instance de `MorphologyParam` avec les champs initialisés aux valeurs données.

**Paramètres**

**radius** (`int`) – Rayon. Rayon de l'ouverture circulaire (disque). Entier supérieur à 1. Par défaut : 1.

**Renvoie**

Nouvelle instance de `MorphologyParam`.

`cdl.computation.image.morphology.compute_white_tophat(src : ImageObj, p : MorphologyParam) → ImageObj`

Compute White Top-Hat with `skimage.morphology.white_tophat()`

**Paramètres**

— **src** – input image object

— **p** – parameters

**Renvoie**

Output image object

`cdl.computation.image.morphology.compute_black_tophat(src : ImageObj, p : MorphologyParam) → ImageObj`

Compute Black Top-Hat with `skimage.morphology.black_tophat()`

**Paramètres**

— **src** – input image object

— **p** – parameters

**Renvoie**

Output image object

`cdl.computation.image.morphology.compute_erosion(src : ImageObj, p : MorphologyParam) → ImageObj`

Compute Erosion with `skimage.morphology.erosion()`

**Paramètres**

— **src** – input image object

— **p** – parameters

**Renvoie**

Output image object

`cdl.computation.image.morphology.compute_dilation(src : ImageObj, p : MorphologyParam) → ImageObj`

Compute Dilation with `skimage.morphology.dilation()`

**Paramètres**

— **src** – input image object

— **p** – parameters

**Renvoie**

Output image object

`cdl.computation.image.morphology.compute_opening(src : ImageObj, p : MorphologyParam) → ImageObj`

Compute morphological opening with `skimage.morphology.opening()`

**Paramètres**

— **src** – input image object

— **p** – parameters

**Renvoie**

Output image object

`cdl.computation.image.morphology.compute_closing(src : ImageObj, p : MorphologyParam) → ImageObj`

Compute morphological closing with `skimage.morphology.closing()`

**Paramètres**

— **src** – input image object

— **p** – parameters

**Renvoie**

Output image object

**Edge detection features****Edges computation module**

**class** `cdl.computation.image.edges.CannyParam`

Canny filter parameters

**sigma**

Ecart-type du filtrage gaussien. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**low\_threshold**

Seuil bas. Borne inférieure pour le seuillage par hystérésis (liaison des contours). Flottant supérieur à 0. Par défaut : 0.1.

**Type**

`guidata.dataset.dataitems.FloatItem`

**high\_threshold**

Seuil haut. Borne supérieure pour le seuillage par hystérésis (liaison des contours). Flottant supérieur à 0. Par défaut : 0.9.

**Type**

`guidata.dataset.dataitems.FloatItem`

**use\_quantiles**

Interprète les seuils bas et haut en tant que quantiles des niveaux des contours, au lieu de valeurs absolues des contours. Si le réglage est actif, alors les seuils doivent être compris entre 0 et 1. Par défaut : True.

**Type**

`guidata.dataset.dataitems.BoolItem`

**mode**

Sélection unique parmi : “reflect”, “constant”, “nearest”, “mirror”, “wrap”. Par défaut : “constant”.

**Type**

`guidata.dataset.dataitems.ChoiceItem`

**cval**

Valeur de remplissage si le mode est constant. Par défaut : 0.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod** `create(sigma : float, low_threshold : float, high_threshold : float, use_quantiles : bool, mode : str, cval : float) → cdl.computation.image.edges.CannyParam`

Renvoie une nouvelle instance de `CannyParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **sigma** (*float*) – Ecart-type du filtrage gaussien. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 1.0.
- **low\_threshold** (*float*) – Seuil bas. Borne inférieure pour le seuillage par hystérésis (liaison des contours). Flottant supérieur à 0. Par défaut : 0.1.
- **high\_threshold** (*float*) – Seuil haut. Borne supérieure pour le seuillage par hystérésis (liaison des contours). Flottant supérieur à 0. Par défaut : 0.9.
- **use\_quantiles** (*bool*) – Interprète les seuils bas et haut en tant que quantiles des niveaux des contours, au lieu de valeurs absolues des contours. Si le réglage est actif, alors les seuils doivent être compris entre 0 et 1. Par défaut : True.
- **mode** (*str*) – Sélection unique parmi : “reflect”, “constant”, “nearest”, “mirror”, “wrap”. Par défaut : “constant”.
- **cval** (*float*) – Valeur de remplissage si le mode est constant. Par défaut : 0.0.

**Renvoie**

Nouvelle instance de `CannyParam`.

`cdl.computation.image.edges.compute_canny(src : ImageObj, p : CannyParam) → ImageObj`

Compute Canny filter with `skimage.feature.canny()`

**Paramètres**

- **src** – input image object
- **p** – parameters

**Renvoie**

Output image object

`cdl.computation.image.edges.compute_roberts(src : ImageObj) → ImageObj`

Compute Roberts filter with `skimage.filters.roberts()`

**Paramètres**

- src** – input image object

**Renvoie**

Output image object

`cdl.computation.image.edges.compute_prewitt(src : ImageObj) → ImageObj`

Compute Prewitt filter with `skimage.filters.prewitt()`

**Paramètres**

- src** – input image object

**Renvoie**

Output image object

`cdl.computation.image.edges.compute_prewitt_h(src : ImageObj) → ImageObj`

Compute horizontal Prewitt filter with `skimage.filters.prewitt_h()`

**Paramètres**

- src** – input image object

**Renvoie**

Output image object

`cdl.computation.image.edges.compute_prewitt_v(src : ImageObj) → ImageObj`

Compute vertical Prewitt filter with `skimage.filters.prewitt_v()`

**Paramètres**

**src** – input image object

**Renvoie**

Output image object

`cdl.computation.image.edges.compute_sobel(src : ImageObj) → ImageObj`

Compute Sobel filter with `skimage.filters.sobel()`

**Paramètres**

**src** – input image object

**Renvoie**

Output image object

`cdl.computation.image.edges.compute_sobel_h(src : ImageObj) → ImageObj`

Compute horizontal Sobel filter with `skimage.filters.sobel_h()`

**Paramètres**

**src** – input image object

**Renvoie**

Output image object

`cdl.computation.image.edges.compute_sobel_v(src : ImageObj) → ImageObj`

Compute vertical Sobel filter with `skimage.filters.sobel_v()`

**Paramètres**

**src** – input image object

**Renvoie**

Output image object

`cdl.computation.image.edges.compute_scharr(src : ImageObj) → ImageObj`

Compute Scharr filter with `skimage.filters.scharr()`

**Paramètres**

**src** – input image object

**Renvoie**

Output image object

`cdl.computation.image.edges.compute_scharr_h(src : ImageObj) → ImageObj`

Compute horizontal Scharr filter with `skimage.filters.scharr_h()`

**Paramètres**

**src** – input image object

**Renvoie**

Output image object

`cdl.computation.image.edges.compute_scharr_v(src : ImageObj) → ImageObj`

Compute vertical Scharr filter with `skimage.filters.scharr_v()`

**Paramètres**

**src** – input image object

**Renvoie**

Output image object

`cdl.computation.image.edges.compute_farid(src : ImageObj) → ImageObj`

Compute Farid filter with `skimage.filters.farid()`

**Paramètres**

**src** – input image object

**Renvoie**

Output image object

`cdl.computation.image.edges.compute_farid_h(src : ImageObj) → ImageObj`Compute horizontal Farid filter with `skimage.filters.farid_h()`**Paramètres****src** – input image object**Renvoie**

Output image object

`cdl.computation.image.edges.compute_farid_v(src : ImageObj) → ImageObj`Compute vertical Farid filter with `skimage.filters.farid_v()`**Paramètres****src** – input image object**Renvoie**

Output image object

`cdl.computation.image.edges.compute_laplace(src : ImageObj) → ImageObj`Compute Laplace filter with `skimage.filters.laplace()`**Paramètres****src** – input image object**Renvoie**

Output image object

**Detection features****Blob detection computation module****class** `cdl.computation.image.detection.GenericDetectionParam`

Generic detection parameters

**threshold**

Seuil relatif. Seuil de détection, relatif à la différence entre le maximum et le minimum des données de l'image. Flottant compris entre 0.1 et 0.9. Par défaut : 0.5.

**Type**`guidata.dataset.dataitems.FloatItem`**classmethod** `create(threshold : float) → cdl.computation.image.detection.GenericDetectionParam`Renvoie une nouvelle instance de `GenericDetectionParam` avec les champs initialisés aux valeurs données.**Paramètres****threshold** (`float`) – Seuil relatif. Seuil de détection, relatif à la différence entre le maximum et le minimum des données de l'image. Flottant compris entre 0.1 et 0.9. Par défaut : 0.5.**Renvoie**Nouvelle instance de `GenericDetectionParam`.**class** `cdl.computation.image.detection.Peak2DDetectionParam`

Peak detection parameters

**threshold**

Seuil relatif. Seuil de détection, relatif à la différence entre le maximum et le minimum des données de l'image. Flottant compris entre 0.1 et 0.9. Par défaut : 0.5.

**Type**`guidata.dataset.dataitems.FloatItem`**size**

Taille du voisinage. Taille de la fenêtre glissante utilisée dans l’algorithme de filtrage maximum/minimum. Entier supérieur à 1, unité : pixels. Par défaut : 10.

**Type**`guidata.dataset.dataitems.IntItem`**create\_rois**

Par défaut : True.

**Type**`guidata.dataset.dataitems.BoolItem`

**classmethod create**(*threshold : float, size : int, create\_rois : bool*) → *cdl.computation.image.detection.Peak2DDetectionParam*

Renvoie une nouvelle instance de *Peak2DDetectionParam* avec les champs initialisés aux valeurs données.

**Paramètres**

- **threshold** (*float*) – Seuil relatif. Seuil de détection, relatif à la différence entre le maximum et le minimum des données de l’image. Flottant compris entre 0.1 et 0.9. Par défaut : 0.5.
- **size** (*int*) – Taille du voisinage. Taille de la fenêtre glissante utilisée dans l’algorithme de filtrage maximum/minimum. Entier supérieur à 1, unité : pixels. Par défaut : 10.
- **create\_rois** (*bool*) – Par défaut : True.

**Renvoie**

Nouvelle instance de *Peak2DDetectionParam*.

`cdl.computation.image.detection.compute_peak_detection`(*image : ImageObj, p : Peak2DDetectionParam*) → *ResultShape* | *None*

Compute 2D peak detection with *cdl.algorithms.image.get\_2d\_peaks\_coords()*

**Paramètres**

- **imageOutput** – input image
- **p** – parameters

**Renvoie**

Peak coordinates

**class** `cdl.computation.image.detection.ContourShapeParam`

Contour shape parameters

**threshold**

Seuil relatif. Seuil de détection, relatif à la différence entre le maximum et le minimum des données de l’image. Flottant compris entre 0.1 et 0.9. Par défaut : 0.5.

**Type**`guidata.dataset.dataitems.FloatItem`**shape**

Forme. Sélection unique parmi : “ellipse”, “circle”, “polygon”. Par défaut : “ellipse”.

**Type**`guidata.dataset.dataitems.ChoiceItem`

**classmethod create**(*threshold : float, shape : str*) → *cdl.computation.image.detection.ContourShapeParam*

Renvoie une nouvelle instance de `ContourShapeParam` avec les champs initialisés aux valeurs données.

#### Paramètres

- **threshold** (*float*) – Seuil relatif. Seuil de détection, relatif à la différence entre le maximum et le minimum des données de l’image. Flottant compris entre 0.1 et 0.9. Par défaut : 0.5.
- **shape** (*str*) – Forme. Sélection unique parmi : “ellipse”, “circle”, “polygon”. Par défaut : “ellipse”.

#### Renvoie

Nouvelle instance de `ContourShapeParam`.

```
cdl.computation.image.detection.compute_contour_shape(image : ImageObj, p : ContourShapeParam)
→ ResultShape | None
```

Compute contour shape fit with `cdl.algorithms.image.get_contour_shapes()`

**class** `cdl.computation.image.detection.BaseBlobParam`

Base class for blob detection parameters

#### **min\_sigma**

min. L’écart-type minimal pour le noyau gaussien. Cette valeur doit être faible pour détecter de petites taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 1.0.

#### Type

`guidata.dataset.dataitems.FloatItem`

#### **max\_sigma**

max. L’écart-type maximal pour le noyau gaussien. Cette valeur doit être élevée pour détecter de grandes taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 30.0.

#### Type

`guidata.dataset.dataitems.FloatItem`

#### **threshold\_rel**

Seuil relatif. Intensité minimale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.2.

#### Type

`guidata.dataset.dataitems.FloatItem`

#### **overlap**

Recouvrement. Si deux taches ont un taux de recouvrement supérieur à ce seuil, alors la plus petite tache est éliminée. Flottant compris entre 0.0 et 1.0. Par défaut : 0.5.

#### Type

`guidata.dataset.dataitems.FloatItem`

```
classmethod create(min_sigma : float, max_sigma : float, threshold_rel : float, overlap : float) →
cdl.computation.image.detection.BaseBlobParam
```

Renvoie une nouvelle instance de `BaseBlobParam` avec les champs initialisés aux valeurs données.

#### Paramètres

- **min\_sigma** (*float*) – min. L’écart-type minimal pour le noyau gaussien. Cette valeur doit être faible pour détecter de petites taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 1.0.
- **max\_sigma** (*float*) – max. L’écart-type maximal pour le noyau gaussien. Cette valeur doit être élevée pour détecter de grandes taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 30.0.
- **threshold\_rel** (*float*) – Seuil relatif. Intensité minimale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.2.

- **overlap** (*float*) – Recouvrement. Si deux taches ont un taux de recouvrement supérieur à ce seuil, alors la plus petite tache est éliminée. Flottant compris entre 0.0 et 1.0. Par défaut : 0.5.

**Renvoie**

Nouvelle instance de *BaseBlobParam*.

**class** `cdl.computation.image.detection.BlobDOGParm`

Blob detection using Difference of Gaussian method

**min\_sigma**

min· L'écart-type minimal pour le noyau gaussien. Cette valeur doit être faible pour détecter de petites taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**max\_sigma**

max· L'écart-type maximal pour le noyau gaussien. Cette valeur doit être élevée pour détecter de grandes taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 30.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**threshold\_rel**

Seuil relatif. Intensité minimale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.2.

**Type**

`guidata.dataset.dataitems.FloatItem`

**overlap**

Recouvrement. Si deux taches ont un taux de recouvrement supérieur à ce seuil, alors la plus petite tache est éliminée. Flottant compris entre 0.0 et 1.0. Par défaut : 0.5.

**Type**

`guidata.dataset.dataitems.FloatItem`

**exclude\_border**

Si le réglage est actif, exclure les taches de la bordure de l'image. Par défaut : True.

**Type**

`guidata.dataset.dataitems.BoolItem`

**classmethod** `create(min_sigma : float, max_sigma : float, threshold_rel : float, overlap : float, exclude_border : bool) → cdl.computation.image.detection.BlobDOGParm`

Renvoie une nouvelle instance de *BlobDOGParm* avec les champs initialisés aux valeurs données.

**Paramètres**

- **min\_sigma** (*float*) – min· L'écart-type minimal pour le noyau gaussien. Cette valeur doit être faible pour détecter de petites taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 1.0.
- **max\_sigma** (*float*) – max· L'écart-type maximal pour le noyau gaussien. Cette valeur doit être élevée pour détecter de grandes taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 30.0.
- **threshold\_rel** (*float*) – Seuil relatif. Intensité minimale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.2.
- **overlap** (*float*) – Recouvrement. Si deux taches ont un taux de recouvrement supérieur à ce seuil, alors la plus petite tache est éliminée. Flottant compris entre 0.0 et 1.0. Par défaut : 0.5.
- **exclude\_border** (*bool*) – Si le réglage est actif, exclure les taches de la bordure de l'image. Par défaut : True.

**Renvoie**

Nouvelle instance de *BlobDOGParam*.

`cdl.computation.image.detection.compute_blob_dog(image : ImageObj, p : BlobDOGParam) →`  
*ResultShape* | None

Compute blobs using Difference of Gaussian method with *cdl.algorithms.image.find\_blobs\_dog()*

**Paramètres**

- **imageOutput** – input image
- **p** – parameters

**Renvoie**

Blobs coordinates

**class** `cdl.computation.image.detection.BlobDOHParam`

Blob detection using Determinant of Hessian method

**min\_sigma**

min. L'écart-type minimal pour le noyau gaussien. Cette valeur doit être faible pour détecter de petites taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 1.0.

**Type**

*guidata.dataset.dataitems.FloatItem*

**max\_sigma**

max. L'écart-type maximal pour le noyau gaussien. Cette valeur doit être élevée pour détecter de grandes taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 30.0.

**Type**

*guidata.dataset.dataitems.FloatItem*

**threshold\_rel**

Seuil relatif. Intensité minimale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.2.

**Type**

*guidata.dataset.dataitems.FloatItem*

**overlap**

Recouvrement. Si deux taches ont un taux de recouvrement supérieur à ce seuil, alors la plus petite tache est éliminée. Flottant compris entre 0.0 et 1.0. Par défaut : 0.5.

**Type**

*guidata.dataset.dataitems.FloatItem*

**log\_scale**

Les valeurs intermédiaires d'écart-type peuvent être interpolées selon une échelle linéaire ou logarithmique. Par défaut : False.

**Type**

*guidata.dataset.dataitems.BoolItem*

**classmethod** `create(min_sigma : float, max_sigma : float, threshold_rel : float, overlap : float, log_scale :`  
*bool*) → *cdl.computation.image.detection.BlobDOHParam*

Renvoie une nouvelle instance de *BlobDOHParam* avec les champs initialisés aux valeurs données.

**Paramètres**

- **min\_sigma** (*float*) – min. L'écart-type minimal pour le noyau gaussien. Cette valeur doit être faible pour détecter de petites taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 1.0.
- **max\_sigma** (*float*) – max. L'écart-type maximal pour le noyau gaussien. Cette valeur doit être élevée pour détecter de grandes taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 30.0.

- **threshold\_rel** (*float*) – Seuil relatif. Intensité minimale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.2.
- **overlap** (*float*) – Recouvrement. Si deux taches ont un taux de recouvrement supérieur à ce seuil, alors la plus petite tache est éliminée. Flottant compris entre 0.0 et 1.0. Par défaut : 0.5.
- **log\_scale** (*bool*) – Les valeurs intermédiaires d'écart-type peuvent être interpolées selon une échelle linéaire ou logarithmique. Par défaut : False.

**Renvoie**

Nouvelle instance de *BlobDOHParam*.

`cdl.computation.image.detection.compute_blob_doh(image : ImageObj, p : BlobDOHParam) → ResultShape | None`

Compute blobs using Determinant of Hessian method with `cdl.algorithms.image.find_blobs_doh()`

**Paramètres**

- **imageOutput** – input image
- **p** – parameters

**Renvoie**

Blobs coordinates

**class** `cdl.computation.image.detection.BlobLOGParam`

Blob detection using Laplacian of Gaussian method

**min\_sigma**

min. L'écart-type minimal pour le noyau gaussien. Cette valeur doit être faible pour détecter de petites taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**max\_sigma**

max. L'écart-type maximal pour le noyau gaussien. Cette valeur doit être élevée pour détecter de grandes taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 30.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**threshold\_rel**

Seuil relatif. Intensité minimale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.2.

**Type**

`guidata.dataset.dataitems.FloatItem`

**overlap**

Recouvrement. Si deux taches ont un taux de recouvrement supérieur à ce seuil, alors la plus petite tache est éliminée. Flottant compris entre 0.0 et 1.0. Par défaut : 0.5.

**Type**

`guidata.dataset.dataitems.FloatItem`

**log\_scale**

Les valeurs intermédiaires d'écart-type peuvent être interpolées selon une échelle linéaire ou logarithmique. Par défaut : False.

**Type**

`guidata.dataset.dataitems.BoolItem`

**exclude\_border**

Si le réglage est actif, exclure les taches de la bordure de l'image. Par défaut : True.

**Type**`guidata.dataset.dataitems.BoolItem`

**classmethod** `create(min_sigma : float, max_sigma : float, threshold_rel : float, overlap : float, log_scale : bool, exclude_border : bool) → cdl.computation.image.detection.BlobLOGParam`

Renvoie une nouvelle instance de `BlobLOGParam` avec les champs initialisés aux valeurs données.

**Paramètres**

- **min\_sigma** (*float*) – <sub>min</sub>. L'écart-type minimal pour le noyau gaussien. Cette valeur doit être faible pour détecter de petites taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 1.0.
- **max\_sigma** (*float*) – <sub>max</sub>. L'écart-type maximal pour le noyau gaussien. Cette valeur doit être élevée pour détecter de grandes taches. Flottant supérieur à 0, non nul, unité : pixels. Par défaut : 30.0.
- **threshold\_rel** (*float*) – Seuil relatif. Intensité minimale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.2.
- **overlap** (*float*) – Recouvrement. Si deux taches ont un taux de recouvrement supérieur à ce seuil, alors la plus petite tache est éliminée. Flottant compris entre 0.0 et 1.0. Par défaut : 0.5.
- **log\_scale** (*bool*) – Les valeurs intermédiaires d'écart-type peuvent être interpolées selon une échelle linéaire ou logarithmique. Par défaut : False.
- **exclude\_border** (*bool*) – Si le réglage est actif, exclure les taches de la bordure de l'image. Par défaut : True.

**Renvoie**

Nouvelle instance de `BlobLOGParam`.

`cdl.computation.image.detection.compute_blob_log(image : ImageObj, p : BlobLOGParam) → ResultShape | None`

Compute blobs using Laplacian of Gaussian method with `cdl.algorithms.image.find_blobs_log()`

**Paramètres**

- **imageOutput** – input image
- **p** – parameters

**Renvoie**

Blobs coordinates

**class** `cdl.computation.image.detection.BlobOpenCVParam`

Blob detection using OpenCV

**min\_threshold**

Seuil min. Le seuil minimum entre les maxima et minima locaux. Ce paramètre n'affecte pas la qualité des taches, mais seulement leur nombre. Un seuil bas donne un nombre plus important de taches. Flottant supérieur à 0.0. Par défaut : 10.0.

**Type**`guidata.dataset.dataitems.FloatItem`**max\_threshold**

Seuil max. Le seuil maximum entre les maxima et minima locaux. Ce paramètre n'affecte pas la qualité des taches, mais seulement leur nombre. Un seuil bas donne un nombre plus important de taches. Flottant supérieur à 0.0. Par défaut : 200.0.

**Type**`guidata.dataset.dataitems.FloatItem`**min\_repeatability**

Répétabilité min. Le nombre minimum de fois qu'une tache doit être détectée dans une séquence d'images pour être considérée valide. Entier supérieur à 1. Par défaut : 2.

**Type**`guidata.dataset.dataitems.IntItem`**min\_dist\_between\_blobs**

Distance min. entre taches. La distance minimale entre deux taches. Si deux taches sont détectées à une distance inférieure à celle-ci, alors la plus petite tache est éliminée. Flottant supérieur à 0.0, non nul. Par défaut : 10.0.

**Type**`guidata.dataset.dataitems.FloatItem`**filter\_by\_color**

Si vrai, l'image est filtrée par couleur au lieu d'intensité. Par défaut : True.

**Type**`guidata.dataset.dataitems.BoolItem`**blob\_color**

Couleur de la tache. La couleur des taches à détecter (0 pour les taches foncées, 255 pour les taches claires). Par défaut : 0.

**Type**`guidata.dataset.dataitems.IntItem`**filter\_by\_area**

Si vrai, l'image est filtrée par l'aire des taches. Par défaut : True.

**Type**`guidata.dataset.dataitems.BoolItem`**min\_area**

Aire min. L'aire minimale des taches. Flottant supérieur à 0.0. Par défaut : 25.0.

**Type**`guidata.dataset.dataitems.FloatItem`**max\_area**

Aire max. L'aire maximale des taches. Flottant supérieur à 0.0. Par défaut : 500.0.

**Type**`guidata.dataset.dataitems.FloatItem`**filter\_by\_circularity**

Si vrai, l'image est filtrée par la circularité des taches. Par défaut : False.

**Type**`guidata.dataset.dataitems.BoolItem`**min\_circularity**

Circularité min. La circularité minimale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.8.

**Type**`guidata.dataset.dataitems.FloatItem`**max\_circularity**

Circularité max. La circularité maximale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 1.0.

**Type**`guidata.dataset.dataitems.FloatItem`**filter\_by\_inertia**

Si vrai, l'image est filtrée par l'inertie des taches. Par défaut : False.

**Type**`guidata.dataset.dataitems.BoolItem`

**min\_inertia\_ratio**

Ratio d'inertie min. Le ratio d'inertie minimal des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.6.

**Type**

`guidata.dataset.dataitems.FloatItem`

**max\_inertia\_ratio**

Ratio d'inertie max. Le ratio d'inertie maximal des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**filter\_by\_convexity**

Si vrai, l'image est filtrée par la convexité des taches. Par défaut : False.

**Type**

`guidata.dataset.dataitems.BoolItem`

**min\_convexity**

Convexité min. La convexité minimale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.8.

**Type**

`guidata.dataset.dataitems.FloatItem`

**max\_convexity**

Convexité max. La convexité maximale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 1.0.

**Type**

`guidata.dataset.dataitems.FloatItem`

**classmethod create**(*min\_threshold* : float, *max\_threshold* : float, *min\_repeatability* : int, *min\_dist\_between\_blobs* : float, *filter\_by\_color* : bool, *blob\_color* : int, *filter\_by\_area* : bool, *min\_area* : float, *max\_area* : float, *filter\_by\_circularity* : bool, *min\_circularity* : float, *max\_circularity* : float, *filter\_by\_inertia* : bool, *min\_inertia\_ratio* : float, *max\_inertia\_ratio* : float, *filter\_by\_convexity* : bool, *min\_convexity* : float, *max\_convexity* : float) → *cdl.computation.image.detection.BlobOpenCVParam*

Renvoie une nouvelle instance de [BlobOpenCVParam](#) avec les champs initialisés aux valeurs données.

**Paramètres**

- **min\_threshold** (float) – Seuil min. Le seuil minimum entre les maxima et minima locaux. Ce paramètre n'affecte pas la qualité des taches, mais seulement leur nombre. Un seuil bas donne un nombre plus important de taches. Flottant supérieur à 0.0. Par défaut : 10.0.
- **max\_threshold** (float) – Seuil max. Le seuil maximum entre les maxima et minima locaux. Ce paramètre n'affecte pas la qualité des taches, mais seulement leur nombre. Un seuil bas donne un nombre plus important de taches. Flottant supérieur à 0.0. Par défaut : 200.0.
- **min\_repeatability** (int) – Répétabilité min. Le nombre minimum de fois qu'une tache doit être détectée dans une séquence d'images pour être considérée valide. Entier supérieur à 1. Par défaut : 2.
- **min\_dist\_between\_blobs** (float) – Distance min. entre taches. La distance minimale entre deux taches. Si deux taches sont détectées à une distance inférieure à celle-ci, alors la plus petite tache est éliminée. Flottant supérieur à 0.0, non nul. Par défaut : 10.0.
- **filter\_by\_color** (bool) – Si vrai, l'image est filtrée par couleur au lieu d'intensité. Par défaut : True.
- **blob\_color** (int) – Couleur de la tache. La couleur des taches à détecter (0 pour les taches foncées, 255 pour les taches claires). Par défaut : 0.

- **filter\_by\_area** (*bool*) – Si vrai, l'image est filtrée par l'aire des taches. Par défaut : True.
- **min\_area** (*float*) – Aire min. L'aire minimale des taches. Flottant supérieur à 0.0. Par défaut : 25.0.
- **max\_area** (*float*) – Aire max. L'aire maximale des taches. Flottant supérieur à 0.0. Par défaut : 500.0.
- **filter\_by\_circularity** (*bool*) – Si vrai, l'image est filtrée par la circularité des taches. Par défaut : False.
- **min\_circularity** (*float*) – Circularité min. La circularité minimale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.8.
- **max\_circularity** (*float*) – Circularité max. La circularité maximale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 1.0.
- **filter\_by\_inertia** (*bool*) – Si vrai, l'image est filtrée par l'inertie des taches. Par défaut : False.
- **min\_inertia\_ratio** (*float*) – Ratio d'inertie min. Le ratio d'inertie minimal des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.6.
- **max\_inertia\_ratio** (*float*) – Ratio d'inertie max. Le ratio d'inertie maximal des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 1.0.
- **filter\_by\_convexity** (*bool*) – Si vrai, l'image est filtrée par la convexité des taches. Par défaut : False.
- **min\_convexity** (*float*) – Convexité min. La convexité minimale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 0.8.
- **max\_convexity** (*float*) – Convexité max. La convexité maximale des taches. Flottant compris entre 0.0 et 1.0. Par défaut : 1.0.

**Renvoie**

Nouvelle instance de *BlobOpenCVParam*.

`cdl.computation.image.detection.compute_blob_opencv(image : ImageObj, p : BlobOpenCVParam) → ResultShape | None`

Compute blobs using OpenCV with *cdl.algorithms.image.find\_blobs\_opencv()*

**Paramètres**

- **imageOutput** – input image
- **p** – parameters

**Renvoie**

Blobs coordinates

## 3.5 Objets proxy (cdl.proxy)

Le module *cdl.proxy* fournit un moyen d'accéder aux fonctionnalités de DataLab à partir d'une classe proxy.

La liste des méthodes de calcul accessibles à partir des objets proxy est disponible dans la section *Appel des méthodes de calcul à l'aide d'objets proxy*.

### 3.5.1 Proxy distant

Le proxy distant est utilisé lorsque DataLab est démarré à partir d'un processus différent du proxy. Dans ce cas, le proxy se connecte au serveur XML-RPC de DataLab.

**class** `cdl.proxy.RemoteProxy`(*autoconnect* : *bool* = *True*)

Classe proxy distant DataLab.

Cette classe fournit un accès aux fonctionnalités de DataLab à partir d'une classe proxy. Il s'agit de la version distante du proxy, qui est utilisée lorsque DataLab est démarré à partir d'un processus différent du proxy.

#### Paramètres

**autoconnect** (*bool*) – Connexion automatique au serveur XML-RPC de DataLab.

#### Lève

- **ConnectionRefusedError** – Impossible de se connecter à DataLab
- **ValueError** – Délai d'attente invalide (doit être  $\geq 0.0$ )
- **ValueError** – Nombre de tentatives invalide (doit être  $\geq 1$ )

---

**Note :** The proxy object also allows to access DataLab computing methods exposed by the processor classes (see *Appel des méthodes de calcul à l'aide d'objets proxy*).

---

### Exemples

Voici un exemple simple de l'utilisation de RemoteProxy dans un script Python ou dans un notebook Jupyter :

```
>>> from cdl.proxy import RemoteProxy
>>> proxy = RemoteProxy()
Connecting to DataLab XML-RPC server...OK (port: 28867)
>>> proxy.get_version()
'1.0.0'
>>> proxy.add_signal("toto", np.array([1., 2., 3.]), np.array([4., 5., -1.]))
True
>>> proxy.get_object_titles()
['toto']
>>> proxy["toto"] # from title
<cdl.core.model.signal.SignalObj at 0x7f7f1c0b4a90>
>>> proxy[1] # from number
<cdl.core.model.signal.SignalObj at 0x7f7f1c0b4a90>
>>> proxy[1].data
array([1., 2., 3.])
>>> proxy.set_current_panel("image")
```

**add\_annotations\_from\_items**(*items* : *list*, *refresh\_plot* : *bool* = *True*, *panel* : *str* | *None* = *None*) → *None*

Ajouter des annotations d'objet (éléments de tracé d'annotation).

#### Paramètres

- **items** – éléments de tracé d'annotation
- **refresh\_plot** – rafraîchir le tracé. Par défaut, *True*.
- **panel** – nom du panneau (valeurs valides : « signal », « image »). Si *None*, le panneau actuel est utilisé.

**add\_image**(*title* : *str*, *data* : *ndarray*, *xunit* : *str* | *None* = *None*, *yunit* : *str* | *None* = *None*, *zunit* : *str* | *None* = *None*, *xlabel* : *str* | *None* = *None*, *ylabel* : *str* | *None* = *None*, *zlabel* : *str* | *None* = *None*) → *bool*

Ajouter des données d'image à DataLab.

**Paramètres**

- **title** – Titre de l'image
- **data** – Données de l'image
- **xunit** – Unité X. Par défaut, None.
- **yunit** – Unité Y. Par défaut, None.
- **zunit** – Unité Z. Par défaut, None.
- **xlabel** – Libellé X. Par défaut, None.
- **ylabel** – Libellé Y. Par défaut, None.
- **zlabel** – Libellé Z. Par défaut, None.

**Renvoie**

Vrai si l'image a été ajoutée avec succès, Faux sinon

**Lève**

**ValueError** – Type de données invalide

**add\_label\_with\_title**(*title* : *str* | *None* = *None*, *panel* : *str* | *None* = *None*) → *None*

Ajouter une étiquette avec le titre de l'objet sur le tracé associé

**Paramètres**

- **title** – Titre de l'étiquette. Par défaut, None. Si None, le titre est le titre de l'objet.
- **panel** – nom du panneau (valeurs valides : « signal », « image »). Si None, le panneau actuel est utilisé.

**add\_object**(*obj* : *SignalObj* | *ImageObj*) → *None*

Ajouter un objet à DataLab.

**Paramètres**

**obj** – Objet signal ou image

**add\_signal**(*title* : *str*, *xdata* : *ndarray*, *ydata* : *ndarray*, *xunit* : *str* | *None* = *None*, *yunit* : *str* | *None* = *None*, *xlabel* : *str* | *None* = *None*, *ylabel* : *str* | *None* = *None*) → *bool*

Ajouter des données de signal à DataLab.

**Paramètres**

- **title** – Titre du signal
- **xdata** – Données X
- **ydata** – Données Y
- **xunit** – Unité X. Par défaut, None.
- **yunit** – Unité Y. Par défaut, None.
- **xlabel** – Libellé X. Par défaut, None.
- **ylabel** – Libellé Y. Par défaut, None.

**Renvoie**

Vrai si le signal a été ajouté avec succès, Faux sinon

**Lève**

- **ValueError** – Type de données xdata invalide
- **ValueError** – Type de données ydata invalide

**calc**(*name* : *str*, *param* : *DataSet* | *None* = *None*) → *None*

Appeler la fonction de calcul name dans le processeur du panneau actuel.

**Paramètres**

- **name** – Nom de la fonction de calcul
- **param** – Paramètre de la fonction de calcul. Par défaut, None.

**Lève**

**ValueError** – fonction inconnue

**close\_application**() → *None*

Fermer l'application DataLab

**connect**(*port* : *str* | *None* = *None*, *timeout* : *float* | *None* = *None*, *retries* : *int* | *None* = *None*) → *None*

Tentative de connexion au serveur XML-RPC de DataLab.

**Paramètres**

- **port** – Port XML-RPC auquel se connecter. S’il n’est pas spécifié, le port est récupéré automatiquement à partir de la configuration de DataLab.
- **timeout** – Délai d’attente en secondes. Par défaut, 5.0.
- **retries** – Nombre de tentatives. Par défaut, 10.

**Lève**

- **ConnectionRefusedError** – Impossible de se connecter à DataLab
- **ValueError** – Délai d’attente invalide (doit être >= 0.0)
- **ValueError** – Nombre de tentatives invalide (doit être >= 1)

**context\_no\_refresh**() → *Generator*[*None*, *None*, *None*]

Renvoie un gestionnaire de contexte pour désactiver temporairement le rafraîchissement automatique.

**Renvoie**

Gestionnaire de contexte

**Exemple**

```
>>> with proxy.context_no_refresh():
...     proxy.add_image("image1", data1)
...     proxy.compute_fft()
...     proxy.compute_wiener()
...     proxy.compute_ifft()
...     # Auto refresh is disabled during the above operations
```

**delete\_metadata**(*refresh\_plot* : *bool* = *True*, *keep\_roi* : *bool* = *False*) → *None*

Supprimer les métadonnées des objets sélectionnés

**Paramètres**

- **refresh\_plot** – Actualiser le tracé. Par défaut, *True*.
- **keep\_roi** – Conserver la ROI. Par défaut, *False*.

**disconnect**() → *None*

Déconnexion du serveur XML-RPC de DataLab.

**get\_current\_panel**() → *str*

Renvoie le nom du panneau actuel.

**Renvoie**

« signal », « image », « macro »))

**Type renvoyé**

Panel name (valid values

**get\_group\_titles\_with\_object\_infos**() → *tuple*[*list*[*str*], *list*[*list*[*str*]], *list*[*list*[*str*]]]

Renvoie les titres des groupes et les listes des UUID et des titres des objets internes.

**Renvoie**

titres des groupes, listes des UUID et des titres des objets internes

**Type renvoyé**

Tuple

**get\_method\_list**() → *list*[*str*]

Renvoie la liste des méthodes disponibles.

**get\_object**(*nb\_id\_title* : *int* | *str* | *None* = *None*, *panel* : *str* | *None* = *None*) → *SignalObj* | *ImageObj*

Obtenir l'objet (signal/image) à partir de l'index.

**Paramètres**

- **nb\_id\_title** – Numéro de l'objet, ou ID de l'objet, ou titre de l'objet. Par défaut, *None* (objet actuel).
- **panel** – Nom du panneau. Par défaut, *None* (panneau actuel).

**Renvoie**

Objet

**Lève**

**KeyError** – si l'objet n'est pas trouvé

**get\_object\_shapes**(*nb\_id\_title* : *int* | *str* | *None* = *None*, *panel* : *str* | *None* = *None*) → *list*

Obtenir les formes des éléments de tracé associées à l'objet (signal/image).

**Paramètres**

- **nb\_id\_title** – Numéro de l'objet, ou ID de l'objet, ou titre de l'objet. Par défaut, *None* (objet actuel).
- **panel** – Nom du panneau. Par défaut, *None* (panneau actuel).

**Renvoie**

Liste des formes des éléments de tracé

**get\_object\_titles**(*panel* : *str* | *None* = *None*) → *list[str]*

Obtenir la liste des objets (signal/image) pour le panneau actuel. Les objets sont triés par numéro de groupe et index d'objet dans le groupe.

**Paramètres**

- panel** – nom du panneau (valeurs valides : « signal », « image », « macro »). Si *None*, le panneau de données actuel est utilisé (c'est-à-dire le panneau signal ou image).

**Renvoie**

Liste des titres des objets

**Lève**

**ValueError** – si le panneau n'est pas trouvé

**get\_object\_uuids**(*panel* : *str* | *None* = *None*) → *list[str]*

Obtenir la liste des UUID des objets (signal/image) pour le panneau actuel. Les objets sont triés par numéro de groupe et index d'objet dans le groupe.

**Paramètres**

- panel** – nom du panneau (valeurs valides : « signal », « image »). Si *None*, le panneau actuel est utilisé.

**Renvoie**

List of object uuids

**Lève**

**ValueError** – si le panneau n'est pas trouvé

**classmethod get\_public\_methods**() → *list[str]*

Renvoie toutes les méthodes publiques de la classe, à l'exception de celle-ci.

**Renvoie**

Liste des méthodes publiques

**get\_sel\_object\_uuids**(*include\_groups* : *bool* = *False*) → *list[str]*

Renvoie les UUID des objets sélectionnés.

**Paramètres**

- include\_groups** – Si *True*, renvoie également les objets des groupes sélectionnés.

**Renvoie**

Liste des UUID des objets sélectionnés.

**get\_version()** → *str*

Renvoie la version publique de DataLab.

**Renvoie**

Version de DataLab

**import\_h5\_file**(*filename* : *str*, *reset\_all* : *bool* | *None* = *None*) → *None*

Ouvrir le navigateur DataLab HDF5 pour importer un fichier HDF5.

**Paramètres**

— **filename** – Nom du fichier HDF5

— **reset\_all** – Réinitialiser toutes les données de l'application. Par défaut, *None*.

**import\_macro\_from\_file**(*filename* : *str*) → *None*

Importer une macro à partir d'un fichier

**Paramètres**

**filename** – Nom de fichier.

**is\_connected()** → *bool*

Renvoie *True* si connecté au serveur XML-RPC de DataLab.

**load\_from\_files**(*filenames* : *list[str]*) → *None*

Ouvrir des objets à partir de fichiers dans le panneau actuel (signaux/images).

**Paramètres**

**filenames** – liste de noms de fichiers

**open\_h5\_files**(*h5files* : *list[str]* | *None* = *None*, *import\_all* : *bool* | *None* = *None*, *reset\_all* : *bool* | *None* = *None*) → *None*

Ouvrir un fichier HDF5 DataLab ou importer à partir de tout autre fichier HDF5.

**Paramètres**

— **h5files** – Liste des fichiers HDF5 à ouvrir. Par défaut, *None*.

— **import\_all** – Importer tous les objets à partir des fichiers HDF5. Par défaut, *None*.

— **reset\_all** – Réinitialiser toutes les données de l'application. Par défaut, *None*.

**raise\_window()** → *None*

Afficher la fenêtre DataLab

**reset\_all()** → *None*

Réinitialiser toutes les données de l'application

**run\_macro**(*number\_or\_title* : *int* | *str* | *None* = *None*) → *None*

Exécute la macro.

**Paramètres**

**number\_or\_title** – Numéro de macro, ou titre de macro. Par défaut, *None* (macro actuelle).

**Lève**

**ValueError** – si la macro n'est pas trouvée

**save\_to\_h5\_file**(*filename* : *str*) → *None*

Enregistrer dans un fichier HDF5 DataLab.

**Paramètres**

**filename** – Nom du fichier HDF5

**select\_groups**(*selection* : *list[int]* | *str* | *None* = *None*, *panel* : *str* | *None* = *None*) → *None*

Sélectionner des groupes dans le panneau actuel.

**Paramètres**

- **selection** – Liste de numéros de groupe (1 à N), ou liste d’UUID de groupe, ou None pour sélectionner tous les groupes. Par défaut, None.
- **panel** – nom du panneau (valeurs valides : « signal », « image »). Si None, le panneau actuel est utilisé. Par défaut, None.

**select\_objects**(*selection* : *list[int | str]*, *panel* : *str | None = None*) → *None*

Sélectionner des objets dans le panneau actuel.

**Paramètres**

- **selection** – Liste de numéros d’objet (1 à N) ou d’UUID à sélectionner
- **panel** – nom du panneau (valeurs valides : « signal », « image »). Si None, le panneau actuel est utilisé. Par défaut, None.

**set\_current\_panel**(*panel* : *str*) → *None*

Passer au panneau.

**Paramètres**

- panel** – Nom du panneau (valeurs valides : « signal », « image », « macro »)

**stop\_macro**(*number\_or\_title* : *int | str | None = None*) → *None*

Arrête la macro.

**Paramètres**

- number\_or\_title** – Numéro de macro, ou titre de macro. Par défaut, None (macro actuelle).

**Lève**

- ValueError** – si la macro n’est pas trouvée

**toggle\_auto\_refresh**(*state* : *bool*) → *None*

Basculer l’état de rafraîchissement automatique.

**Paramètres**

- state** – État de rafraîchissement automatique

**toggle\_show\_titles**(*state* : *bool*) → *None*

Basculer l’état d’affichage des titres.

**Paramètres**

- state** – État d’affichage des titres

### 3.5.2 Proxy local

Le proxy local est utilisé lorsque DataLab est démarré à partir du même processus que le proxy. Dans ce cas, le proxy est directement connecté à l’instance de la fenêtre principale de DataLab. Le cas d’utilisation typique est le script de haut niveau.

**class** `cdl.proxy.LocalProxy`(*cdl* : `CDLMainWindow` | `ServerProxy` | *None = None*)

Classe de proxy local DataLab.

Cette classe permet d’accéder aux fonctionnalités de DataLab à partir d’une classe proxy. Il s’agit de la version locale du proxy, utilisée lorsque DataLab est démarré à partir du même processus que le proxy.

**Paramètres**

- cdl** (`CDLMainWindow`) – Instance de `CDLMainWindow`.

---

**Note :** The proxy object also allows to access DataLab computing methods exposed by the processor classes (see *Appel des méthodes de calcul à l’aide d’objets proxy*).

---

**add\_signal**(*title* : *str*, *xdata* : *ndarray*, *ydata* : *ndarray*, *xunit* : *str* | *None* = *None*, *yunit* : *str* | *None* = *None*, *xlabel* : *str* | *None* = *None*, *ylabel* : *str* | *None* = *None*) → *bool*

Ajouter des données de signal à DataLab.

**Paramètres**

- **title** (*str*) – Titre du signal
- **xdata** (*numpy.ndarray*) – Données X
- **ydata** (*numpy.ndarray*) – Données Y
- **xunit** (*str* | *None*) – Unité X. Par défaut, *None*.
- **yunit** (*str* | *None*) – Unité Y. Par défaut, *None*.
- **xlabel** (*str* | *None*) – Libellé X. Par défaut, *None*.
- **ylabel** (*str* | *None*) – Libellé Y. Par défaut, *None*.

**Renvoie**

Vrai si le signal a été ajouté avec succès, Faux sinon

**Type renvoyé**

*bool*

**Lève**

- **ValueError** – Type de données xdata invalide
- **ValueError** – Type de données ydata invalide

**add\_image**(*title* : *str*, *data* : *ndarray*, *xunit* : *str* | *None* = *None*, *yunit* : *str* | *None* = *None*, *zunit* : *str* | *None* = *None*, *xlabel* : *str* | *None* = *None*, *ylabel* : *str* | *None* = *None*, *zlabel* : *str* | *None* = *None*) → *bool*

Ajouter des données d'image à DataLab.

**Paramètres**

- **title** (*str*) – Titre de l'image
- **data** (*numpy.ndarray*) – Données de l'image
- **xunit** (*str* | *None*) – Unité X. Par défaut, *None*.
- **yunit** (*str* | *None*) – Unité Y. Par défaut, *None*.
- **zunit** (*str* | *None*) – Unité Z. Par défaut, *None*.
- **xlabel** (*str* | *None*) – Libellé X. Par défaut, *None*.
- **ylabel** (*str* | *None*) – Libellé Y. Par défaut, *None*.
- **zlabel** (*str* | *None*) – Libellé Z. Par défaut, *None*.

**Renvoie**

Vrai si l'image a été ajoutée avec succès, Faux sinon

**Type renvoyé**

*bool*

**Lève**

- **ValueError** – Type de données invalide

**add\_object**(*obj* : *SignalObj* | *ImageObj*) → *None*

Ajouter un objet à DataLab.

**Paramètres**

- **obj** (*SignalObj* | *ImageObj*) – Objet signal ou image

**calc**(*name* : *str*, *param* : *DataSet* | *None* = *None*) → *None*

Appeler la fonction de calcul *name* dans le processeur du panneau actuel.

**Paramètres**

- **name** – Nom de la fonction de calcul
- **param** – Paramètre de la fonction de calcul. Par défaut, *None*.

**Lève**

- **ValueError** – fonction inconnue

**get\_object**(*nb\_id\_title* : *int* | *str* | *None* = *None*, *panel* : *str* | *None* = *None*) → *SignalObj* | *ImageObj*

Obtenir l'objet (signal/image) à partir de l'index.

**Paramètres**

- **nb\_id\_title** – Numéro de l’objet, ou ID de l’objet, ou titre de l’objet. Par défaut, None (objet actuel).
- **panel** – Nom du panneau. Par défaut, None (panneau actuel).

**Renvoie**

Objet

**Lève****KeyError** – si l’objet n’est pas trouvé**get\_object\_shapes**(nb\_id\_title : *int* | *str* | *None* = *None*, panel : *str* | *None* = *None*) → *list*

Obtenir les formes des éléments de tracé associées à l’objet (signal/image).

**Paramètres**

- **nb\_id\_title** – Numéro de l’objet, ou ID de l’objet, ou titre de l’objet. Par défaut, None (objet actuel).
- **panel** – Nom du panneau. Par défaut, None (panneau actuel).

**Renvoie**

Liste des formes des éléments de tracé

**add\_annotations\_from\_items**(items : *list*, refresh\_plot : *bool* = *True*, panel : *str* | *None* = *None*) → *None*

Ajouter des annotations d’objet (éléments de tracé d’annotation).

**Paramètres**

- **items** (*list*) – éléments de tracé d’annotation
- **refresh\_plot** (*bool* | *None*) – rafraîchir le tracé. Par défaut, *True*.
- **panel** (*str* | *None*) – nom du panneau (valeurs valides : « signal », « image »). Si *None*, le panneau actuel est utilisé.

**add\_label\_with\_title**(title : *str* | *None* = *None*, panel : *str* | *None* = *None*) → *None*

Ajouter une étiquette avec le titre de l’objet sur le tracé associé

**Paramètres**

- **title** – Titre de l’étiquette. Par défaut, *None*. Si *None*, le titre est le titre de l’objet.
- **panel** – nom du panneau (valeurs valides : « signal », « image »). Si *None*, le panneau actuel est utilisé.

**close\_application**() → *None*

Fermer l’application DataLab

**context\_no\_refresh**() → *Generator*[*None*, *None*, *None*]

Renvoie un gestionnaire de contexte pour désactiver temporairement le rafraîchissement automatique.

**Renvoie**

Gestionnaire de contexte

**Exemple**

```
>>> with proxy.context_no_refresh():
...     proxy.add_image("image1", data1)
...     proxy.compute_fft()
...     proxy.compute_wiener()
...     proxy.compute_ifft()
...     # Auto refresh is disabled during the above operations
```

**delete\_metadata**(refresh\_plot : *bool* = *True*, keep\_roi : *bool* = *False*) → *None*

Supprimer les métadonnées des objets sélectionnés

**Paramètres**

- **refresh\_plot** – Actualiser le tracé. Par défaut, True.
- **keep\_roi** – Conserver la ROI. Par défaut, False.

**get\_current\_panel()** → *str*

Renvoie le nom du panneau actuel.

**Renvoie**

« signal », « image », « macro »))

**Type renvoyé**

Panel name (valid values

**get\_group\_titles\_with\_object\_infos()** → *tuple[list[str], list[list[str]], list[list[str]]]*

Renvoie les titres des groupes et les listes des UUID et des titres des objets internes.

**Renvoie**

titres des groupes, listes des UUID et des titres des objets internes

**Type renvoyé**

Tuple

**get\_object\_titles**(*panel : str | None = None*) → *list[str]*

Obtenir la liste des objets (signal/image) pour le panneau actuel. Les objets sont triés par numéro de groupe et index d'objet dans le groupe.

**Paramètres**

**panel** – nom du panneau (valeurs valides : « signal », « image », « macro »). Si None, le panneau de données actuel est utilisé (c'est-à-dire le panneau signal ou image).

**Renvoie**

Liste des titres des objets

**Lève**

**ValueError** – si le panneau n'est pas trouvé

**get\_object\_uuids**(*panel : str | None = None*) → *list[str]*

Obtenir la liste des UUID des objets (signal/image) pour le panneau actuel. Les objets sont triés par numéro de groupe et index d'objet dans le groupe.

**Paramètres**

**panel** – nom du panneau (valeurs valides : « signal », « image »). Si None, le panneau actuel est utilisé.

**Renvoie**

List of object uuids

**Lève**

**ValueError** – si le panneau n'est pas trouvé

**classmethod get\_public\_methods()** → *list[str]*

Renvoie toutes les méthodes publiques de la classe, à l'exception de celle-ci.

**Renvoie**

Liste des méthodes publiques

**get\_sel\_object\_uuids**(*include\_groups : bool = False*) → *list[str]*

Renvoie les UUID des objets sélectionnés.

**Paramètres**

**include\_groups** – Si True, renvoie également les objets des groupes sélectionnés.

**Renvoie**

Liste des UUID des objets sélectionnés.

**get\_version()** → *str*

Renvoie la version publique de DataLab.

**Renvoie**

Version de DataLab

**import\_h5\_file**(filename : *str*, reset\_all : *bool* | *None* = *None*) → *None*

Ouvrir le navigateur DataLab HDF5 pour importer un fichier HDF5.

**Paramètres**

- **filename** – Nom du fichier HDF5
- **reset\_all** – Réinitialiser toutes les données de l'application. Par défaut, *None*.

**import\_macro\_from\_file**(filename : *str*) → *None*

Importer une macro à partir d'un fichier

**Paramètres****filename** – Nom de fichier.**load\_from\_files**(filenames : *list[str]*) → *None*

Ouvrir des objets à partir de fichiers dans le panneau actuel (signaux/images).

**Paramètres****filenames** – liste de noms de fichiers**open\_h5\_files**(h5files : *list[str]* | *None* = *None*, import\_all : *bool* | *None* = *None*, reset\_all : *bool* | *None* = *None*) → *None*

Ouvrir un fichier HDF5 DataLab ou importer à partir de tout autre fichier HDF5.

**Paramètres**

- **h5files** – Liste des fichiers HDF5 à ouvrir. Par défaut, *None*.
- **import\_all** – Importer tous les objets à partir des fichiers HDF5. Par défaut, *None*.
- **reset\_all** – Réinitialiser toutes les données de l'application. Par défaut, *None*.

**raise\_window**() → *None*

Afficher la fenêtre DataLab

**reset\_all**() → *None*

Réinitialiser toutes les données de l'application

**run\_macro**(number\_or\_title : *int* | *str* | *None* = *None*) → *None*

Exécute la macro.

**Paramètres****number\_or\_title** – Numéro de macro, ou titre de macro. Par défaut, *None* (macro actuelle).**Lève****ValueError** – si la macro n'est pas trouvée**save\_to\_h5\_file**(filename : *str*) → *None*

Enregistrer dans un fichier HDF5 DataLab.

**Paramètres****filename** – Nom du fichier HDF5**select\_groups**(selection : *list[int]* | *str* | *None* = *None*, panel : *str* | *None* = *None*) → *None*

Sélectionner des groupes dans le panneau actuel.

**Paramètres**

- **selection** – Liste de numéros de groupe (1 à N), ou liste d'UUID de groupe, ou *None* pour sélectionner tous les groupes. Par défaut, *None*.
- **panel** – nom du panneau (valeurs valides : « signal », « image »). Si *None*, le panneau actuel est utilisé. Par défaut, *None*.

**select\_objects**(selection : *list[int]* | *str*, panel : *str* | *None* = *None*) → *None*

Sélectionner des objets dans le panneau actuel.

**Paramètres**

- **selection** – Liste de numéros d’objet (1 à N) ou d’UUID à sélectionner
- **panel** – nom du panneau (valeurs valides : « signal », « image »). Si None, le panneau actuel est utilisé. Par défaut, None.

**set\_current\_panel**(*panel* : *str*) → *None*

Passer au panneau.

**Paramètres**

- panel** – Nom du panneau (valeurs valides : « signal », « image », « macro »)

**stop\_macro**(*number\_or\_title* : *int* | *str* | *None* = *None*) → *None*

Arrête la macro.

**Paramètres**

- number\_or\_title** – Numéro de macro, ou titre de macro. Par défaut, None (macro actuelle).

**Lève**

- ValueError** – si la macro n’est pas trouvée

**toggle\_auto\_refresh**(*state* : *bool*) → *None*

Basculer l’état de rafraîchissement automatique.

**Paramètres**

- state** – État de rafraîchissement automatique

**toggle\_show\_titles**(*state* : *bool*) → *None*

Basculer l’état d’affichage des titres.

**Paramètres**

- state** – État d’affichage des titres

### 3.5.3 Gestionnaire de contexte proxy

Le gestionnaire de contexte proxy est un moyen pratique de gérer la création et la destruction du proxy. Il est utilisé comme suit :

```
with proxy_context("local") as proxy:
    proxy.add_signal(...)
```

Le type de proxy peut être « local » ou « remote ». Pour un proxy distant, le port peut être spécifié comme « remote :port ».

---

**Note :** Le gestionnaire de contexte proxy permet d’utiliser le proxy dans différents contextes (script Python, notebook Jupyter, etc.). Il permet également de passer sans heurts du proxy local au proxy distant, en conservant le même code à l’intérieur du contexte.

---

**cdl.proxy.proxy\_context**(*what* : *str*) → *Generator*[*LocalProxy* | *RemoteProxy*, *None*, *None*]

Gestionnaire de contexte gérant la création et la destruction du proxy CDL.

**Paramètres**

- what** (*str*) – type de proxy (« local » ou « remote »). Pour un proxy distant, le port peut être spécifié comme « remote :port ».

**Renvoie**

*Générateur*[*LocalProxy* | *RemoteProxy*, *None*, *None*] –

**proxy**

*LocalProxy* si *what* == « local », *RemoteProxy* si *what* == « remote » ou « remote :port »

### Exemple

avec `proxy_context(« local »)` en tant que proxy :  
`proxy.add_signal(...)`

### 3.5.4 Appel des méthodes de calcul à l'aide d'objets proxy

Tous les objets proxy permettent d'accéder aux méthodes de calcul exposées par les classes de processeur de DataLab :

- `cdl.core.gui.processor.signal.SignalProcessor`
- `cdl.core.gui.processor.image.ImageProcessor`

**Voir aussi :**

La liste des méthodes de calcul est disponible dans les tableaux ci-dessous.

Il existe deux façons d'appeler une méthode de calcul :

1. En utilisant la méthode `calc()` de l'objet proxy :

```
# Call a method without parameter
proxy.calc("compute_average")

# This is equivalent to:
proxy.calc("average")

# Call a method with parameters
p = cdl.param.MovingAverageParam.create(n=30)
proxy.calc("compute_moving_average", p)
```

2. En appelant directement la méthode de calcul du processeur à partir de l'objet proxy :

```
# Call a method without parameter
proxy.compute_average()

# Call a method with parameters
p = cdl.param.MovingAverageParam.create(n=30)
proxy.compute_moving_average(p)
```

**Avertissement :** Les méthodes `compute_{name}` ne sont pas définies statiquement dans les classes proxy (et même pas dynamiquement). Elles sont néanmoins disponibles via les objets proxy grâce à la méthode magique `__getattr__()` qui transfère l'appel à la méthode `calc()`. Cependant, cela signifie que les méthodes ne sont pas répertoriées dans la documentation des classes proxy, et elles ne sont pas disponibles dans la fonction d'auto-complétion de votre IDE.

## Nombre de méthodes de calcul

TABLEAU 1 – Nombre de méthodes de calcul

Signal	Image	Total
59	100	159

## Traitement du signal

Le tableau suivant répertorie les méthodes de traitement du signal - il est généré automatiquement à partir du code source :

TABLEAU 2: Méthodes de traitement du signal

Méthode de calcul	Description
<code>compute_abs()</code>	Compute absolute value with <code>cdl.computation.signal.compute_abs()</code>
<code>compute_addition_constant()</code>	Compute sum with a constant
<code>compute_arithmetic()</code>	Compute arithmetic operation between two signals
<code>compute_astype()</code>	Convert data type with <code>cdl.computation.signal.compute_astype()</code>
<code>compute_average()</code>	Compute average with <code>cdl.computation.signal.compute_addition()</code>
<code>compute_bandpass()</code>	Compute band-pass filter
<code>compute_bandstop()</code>	Compute band-stop filter
<code>compute_bandwidth_3db()</code>	Compute bandwidth at -3dB
<code>compute_calibration()</code>	Compute data linear calibration
<code>compute_clip()</code>	Compute maximum data clipping
<code>compute_contrast()</code>	Compute contrast with <code>cdl.computation.signal.compute_contrast()</code>
<code>compute_convolution()</code>	Compute convolution
<code>compute_derivative()</code>	Compute derivative
<code>compute_detrending()</code>	Compute detrending
<code>compute_difference()</code>	Compute difference between two signals
<code>compute_difference_constant()</code>	Compute difference with a constant
<code>compute_division()</code>	Compute division between two signals
<code>compute_division_constant()</code>	Compute division by a constant
<code>compute_dynamic_parameters()</code>	Compute Dynamic Parameters (ENOB, SINAD, THD, SFDR, SNR)
<code>compute_exp()</code>	Compute Log10 with <code>cdl.computation.signal.compute_exp()</code>
<code>compute_fft()</code>	Compute FFT with <code>cdl.computation.signal.compute_fft()</code>
<code>compute_fit()</code>	Compute fitting curve using an interactive dialog
<code>compute_fw1e2()</code>	Compute FW at $1/e^2$ with <code>cdl.computation.signal.compute_fw1e2()</code>
<code>compute_fwhm()</code>	Compute FWHM with <code>cdl.computation.signal.compute_fwhm()</code>
<code>compute_gaussian_filter()</code>	Compute gaussian filter
<code>compute_highpass()</code>	Compute high-pass filter
<code>compute_histogram()</code>	Compute histogram
<code>compute_ifft()</code>	Compute iFFT with <code>cdl.computation.signal.compute_ifft()</code>

suite sur la page suivante

Tableau 2 – suite de la page précédente

Méthode de calcul	Description
<code>compute_im()</code>	Compute imaginary part with <code>cdl.computation.signal.compute_im()</code>
<code>compute_integral()</code>	Compute integral with <code>cdl.computation.signal.compute_integral()</code>
<code>compute_interpolation()</code>	Compute interpolation
<code>compute_log10()</code>	Compute Log10 with <code>cdl.computation.signal.compute_log10()</code>
<code>compute_lowpass()</code>	Compute high-pass filter
<code>compute_magnitude_spectrum()</code>	Compute magnitude spectrum
<code>compute_moving_average()</code>	Compute moving average
<code>compute_moving_median()</code>	Compute moving median
<code>compute_multigaussianfit()</code>	Compute multi-Gaussian fitting curve using an interactive dialog
<code>compute_normalize()</code>	Normalize data with <code>cdl.computation.signal.compute_normalize()</code>
<code>compute_offset_correction()</code>	Compute offset correction
<code>compute_peak_detection()</code>	Detect peaks from data
<code>compute_phase_spectrum()</code>	Compute phase spectrum
<code>compute_polyfit()</code>	Compute polynomial fitting curve
<code>compute_power()</code>	Compute power with <code>cdl.computation.signal.compute_power()</code>
<code>compute_product()</code>	Compute product with <code>cdl.computation.signal.compute_product()</code>
<code>compute_product_constant()</code>	Compute product with a constant
<code>compute_psd()</code>	Compute power spectral density
<code>compute_quadratic_difference()</code>	Compute quadratic difference between two signals
<code>compute_re()</code>	Compute real part with <code>cdl.computation.signal.compute_re()</code>
<code>compute_resampling()</code>	Compute resampling
<code>compute_reverse_x()</code>	Reverse X axis with <code>cdl.computation.signal.compute_reverse_x()</code>
<code>compute_roi_extraction()</code>	Extract Region Of Interest (ROI) from data
<code>compute_sampling_rate_period()</code>	Compute sampling rate and period (mean and std)
<code>compute_sqrt()</code>	Compute square root with <code>cdl.computation.signal.compute_sqrt()</code>
<code>compute_stats()</code>	Compute data statistics
<code>compute_sum()</code>	Compute sum with <code>cdl.computation.signal.compute_addition()</code>
<code>compute_swap_axes()</code>	Swap data axes with <code>cdl.computation.signal.compute_swap_axes()</code>
<code>compute_wiener()</code>	Compute Wiener filter
<code>compute_windowing()</code>	Compute windowing
<code>compute_x_at_minmax()</code>	Compute x at min/max

## Traitement d'image

Le tableau suivant répertorie les méthodes de traitement d'image - il est généré automatiquement à partir du code source :

TABLEAU 3: Méthodes de traitement d'image

Méthode de calcul	Description
<code>compute_abs()</code>	Compute absolute value with <code>cdl.computation.image.compute_abs()</code>
<code>compute_addition_constant()</code>	Compute sum with a constant
<code>compute_adjust_gamma()</code>	Compute gamma correction
<code>compute_adjust_log()</code>	Compute log correction
<code>compute_adjust_sigmoid()</code>	Compute sigmoid correction
<code>compute_all_denoise()</code>	Compute all denoising filters
<code>compute_all_edges()</code>	Compute all edges filters
<code>compute_all_morphology()</code>	Compute all morphology filters
<code>compute_all_threshold()</code>	Compute all threshold algorithms
<code>compute_arithmetic()</code>	Compute arithmetic operation between two images
<code>compute_astype()</code>	Convert data type with <code>cdl.computation.image.compute_astype()</code>
<code>compute_average()</code>	Compute average with <code>cdl.computation.image.compute_addition()</code>
<code>compute_average_profile()</code>	Compute average profile
<code>compute_binning()</code>	Binning image with <code>cdl.computation.image.compute_binning()</code>
<code>compute_black_tophat()</code>	Compute Black Top-Hat
<code>compute_blob_dog()</code>	Compute blob detection using Difference of Gaussian method
<code>compute_blob_doh()</code>	Compute blob detection using Determinant of Hessian method
<code>compute_blob_log()</code>	Compute blob detection using Laplacian of Gaussian method
<code>compute_blob_opencv()</code>	Compute blob detection using OpenCV
<code>compute_butterworth()</code>	Compute Butterworth filter
<code>compute_calibration()</code>	Compute data linear calibration
<code>compute_canny()</code>	Compute Canny filter
<code>compute_centroid()</code>	Compute image centroid
<code>compute_clip()</code>	Compute maximum data clipping
<code>compute_closing()</code>	Compute morphological closing
<code>compute_contour_shape()</code>	Compute contour shape fit
<code>compute_denoise_bilateral()</code>	Compute bilateral filter denoising
<code>compute_denoise_tophat()</code>	Denoise using White Top-Hat
<code>compute_denoise_tv()</code>	Compute Total Variation denoising
<code>compute_denoise_wavelet()</code>	Compute Wavelet denoising
<code>compute_difference()</code>	Compute difference between two images
<code>compute_difference_constant()</code>	Compute difference with a constant
<code>compute_dilation()</code>	Compute Dilation
<code>compute_division()</code>	Compute division between two images
<code>compute_division_constant()</code>	Compute division by a constant
<code>compute_enclosing_circle()</code>	Compute minimum enclosing circle
<code>compute_equalize_adapthist()</code>	Adaptive histogram equalization
<code>compute_equalize_hist()</code>	Histogram equalization
<code>compute_erosion()</code>	Compute Erosion
<code>compute_exp()</code>	Compute Log10 with <code>cdl.computation.image.compute_exp()</code>
<code>compute_farid()</code>	Compute Farid filter

suite sur la page suivante

Tableau 3 – suite de la page précédente

Méthode de calcul	Description
<code>compute_farid_h()</code>	Compute Farid filter (horizontal)
<code>compute_farid_v()</code>	Compute Farid filter (vertical)
<code>compute_fft()</code>	Compute FFT with <code>cdl.computation.image.compute_fft()</code>
<code>compute_flatfield()</code>	Compute flat field correction
<code>compute_fliph()</code>	Flip data horizontally
<code>compute_flipv()</code>	Flip data vertically with <code>cdl.computation.image.compute_flipv()</code>
<code>compute_gaussian_filter()</code>	Compute gaussian filter
<code>compute_histogram()</code>	Compute histogram with <code>cdl.computation.image.compute_histogram()</code>
<code>compute_hough_circle_peaks()</code>	Compute peak detection based on a circle Hough transform
<code>compute_ifft()</code>	Compute iFFT with <code>cdl.computation.image.compute_ifft()</code>
<code>compute_im()</code>	Compute imaginary part with <code>cdl.computation.image.compute_im()</code>
<code>compute_laplace()</code>	Compute Laplace filter
<code>compute_line_profile()</code>	Compute profile along a vertical or horizontal line
<code>compute_log10()</code>	Compute Log10 with <code>cdl.computation.image.compute_log10()</code>
<code>compute_logp1()</code>	Compute base 10 logarithm
<code>compute_magnitude_spectrum()</code>	Compute magnitude spectrum
<code>compute_moving_average()</code>	Compute moving average
<code>compute_moving_median()</code>	Compute moving median
<code>compute_normalize()</code>	Normalize data with <code>cdl.computation.image.compute_normalize()</code>
<code>compute_offset_correction()</code>	Compute offset correction
<code>compute_opening()</code>	Compute morphological opening
<code>compute_peak_detection()</code>	Compute 2D peak detection
<code>compute_phase_spectrum()</code>	Compute phase spectrum
<code>compute_prewitt()</code>	Compute Prewitt filter
<code>compute_prewitt_h()</code>	Compute Prewitt filter (horizontal)
<code>compute_prewitt_v()</code>	Compute Prewitt filter (vertical)
<code>compute_product()</code>	Compute product with <code>cdl.computation.image.compute_product()</code>
<code>compute_product_constant()</code>	Compute product with a constant
<code>compute_psd()</code>	Compute Power Spectral Density (PSD)
<code>compute_quadratic_difference()</code>	Compute quadratic difference between two images
<code>compute_radial_profile()</code>	Compute radial profile
<code>compute_re()</code>	Compute real part with <code>cdl.computation.image.compute_re()</code>
<code>compute_rescale_intensity()</code>	Rescale image intensity levels
<code>compute_resize()</code>	Resize image with <code>cdl.computation.image.compute_resize()</code>
<code>compute_roberts()</code>	Compute Roberts filter
<code>compute_roi_extraction()</code>	Extract Region Of Interest (ROI) from data
<code>compute_rotate()</code>	Rotate data arbitrarily
<code>compute_rotate270()</code>	Rotate data 270° with <code>cdl.computation.image.compute_rotate270()</code>
<code>compute_rotate90()</code>	Rotate data 90° with <code>cdl.computation.image.compute_rotate90()</code>
<code>compute_scharr()</code>	Compute Scharr filter

suite sur la page suivante

Tableau 3 – suite de la page précédente

Méthode de calcul	Description
<code>compute_scharr_h()</code>	Compute Scharr filter (horizontal)
<code>compute_scharr_v()</code>	Compute Scharr filter (vertical)
<code>compute_segment_profile()</code>	Compute profile along a segment
<code>compute_sobel()</code>	Compute Sobel filter
<code>compute_sobel_h()</code>	Compute Sobel filter (horizontal)
<code>compute_sobel_v()</code>	Compute Sobel filter (vertical)
<code>compute_stats()</code>	Compute data statistics
<code>compute_sum()</code>	Compute sum with <code>cdl.computation.image.compute_addition()</code>
<code>compute_swap_axes()</code>	Swap data axes with <code>cdl.computation.image.compute_swap_axes()</code>
<code>compute_threshold()</code>	Compute parametric threshold
<code>compute_threshold_isodata()</code>	Compute threshold using Isodata algorithm
<code>compute_threshold_li()</code>	Compute threshold using Li algorithm
<code>compute_threshold_mean()</code>	Compute threshold using Mean algorithm
<code>compute_threshold_minimum()</code>	Compute threshold using Minimum algorithm
<code>compute_threshold_otsu()</code>	Compute threshold using Otsu algorithm
<code>compute_threshold_triangle()</code>	Compute threshold using Triangle algorithm
<code>compute_threshold_yen()</code>	Compute threshold using Yen algorithm
<code>compute_white_tophat()</code>	Compute White Top-Hat
<code>compute_wiener()</code>	Compute Wiener filter

## 3.6 GUI

Le paquet `cdl.core.gui` contient des fonctionnalités liées à l'interface graphique (GUI) du projet DataLab (CDL). Ces fonctionnalités sont principalement spécifiques à DataLab et ne sont pas destinées à être utilisées indépendamment.

Le but de cette section de la documentation est de fournir un aperçu de l'architecture de l'interface graphique de DataLab et de décrire les principales fonctionnalités des modules contenus dans ce paquet. Il ne vise pas à fournir une description détaillée des fonctionnalités de l'interface graphique, mais plutôt à fournir un point de départ pour le lecteur qui souhaite comprendre l'architecture interne de DataLab.

La fenêtre principale de DataLab est composée de plusieurs parties, chacune d'elles étant gérée par un module spécifique de ce paquet :

- Les **Panneaux Signal et Image** : ces panneaux sont utilisés pour afficher des signaux et des images et pour fournir un ensemble d'outils pour les manipuler. Chaque panneau de données repose sur un ensemble de modules pour gérer les fonctionnalités de l'interface graphique (`cdl.core.gui.actionhandler` et `cdl.core.gui.objectview`), le modèle de données (`cdl.core.gui.objectmodel`), la visualisation des données (`cdl.core.gui.plothandler`), et le traitement des données (`cdl.core.gui.processor`).
- Le **Panneau Macro** : ce panneau est utilisé pour afficher et exécuter des macros. Il repose sur le module `cdl.core.gui.macroeditor` pour gérer l'édition et l'exécution des macros.
- Les **widgets spécialisés** : ces widgets sont utilisés pour gérer des fonctionnalités spécifiques telles que l'édition des ROI (`cdl.core.gui.roieditor`), l'édition des profils d'intensité (`cdl.core.gui.profiledialog`), etc.

Sous-module	Objectif
<code>cdl.core.gui.main</code>	Fenêtre principale et application DataLab
<code>cdl.core.gui.panel</code>	Panneaux Signal, Image et Macro
<code>cdl.core.gui.actionhandler</code>	Actions de l'application (menus, barres d'outils, menu contextuel)
<code>cdl.core.gui.objectview</code>	Widgets pour afficher les arbres d'objets (signal/image)
<code>cdl.core.gui.plothandler</code>	Items de visualisation <i>PlotPy</i> pour représenter des signaux et des images
<code>cdl.core.gui.roieditor</code>	Editeur de ROI
<code>cdl.core.gui.processor</code>	Processeur
<code>cdl.core.gui.docks</code>	Widgets dockables
<code>cdl.core.gui.h5io</code>	Entrée/sortie HDF5

## 3.7 Fenêtre principale

Le module `cdl.core.gui.main` fournit la fenêtre principale du projet DataLab (CDL).

**class** `cdl.core.gui.main.CDLMainWindow`(*console=None, hide\_on\_close=False*)

Fenêtre principale de DataLab

### Paramètres

- **console** – activer la console interne
- **hide\_on\_close** – True pour masquer la fenêtre à la fermeture

**static** `get_instance`(*console=None, hide\_on\_close=False*)

Retourne l'instance singleton

**static** `xmlrpc_server_started`(*port*)

Le serveur XML-RPC a démarré, écriture du port de communication dans le fichier de configuration

**get\_group\_titles\_with\_object\_infos**() → `tuple[list[str], list[list[str]], list[list[str]]]`

Retourne les titres des groupes et les listes des uuids et titres des objets internes.

### Renvoie

Groups titles, lists of inner objects uuids and titles

**get\_object\_titles**(*panel : str | None = None*) → `list[str]`

Get object (signal/image) list for current panel. Objects are sorted by group number and object index in group.

### Paramètres

**panel** – panel name (valid values : « signal », « image », « macro »). If None, current data panel is used (i.e. signal or image panel).

### Renvoie

List of object titles

### Lève

**ValueError** – if panel is unknown

**get\_object**(*nb\_id\_title : int | str | None = None, panel : str | None = None*) → `SignalObj | ImageObj`

Get object (signal/image) from index.

### Paramètres

- **nb\_id\_title** – Object number, or object id, or object title. Defaults to None (current object).
- **panel** – Panel name. Defaults to None (current panel).

### Renvoie

Object

**Lève**

- **KeyError** – if object not found
- **TypeError** – if index\_id\_title type is invalid

**get\_object\_uuids**(*panel* : *str* | *None* = *None*) → *list*[*str*]

Get object (signal/image) uuid list for current panel. Objects are sorted by group number and object index in group.

**Paramètres**

**panel** – panel name (valid values : « signal », « image »). If *None*, current panel is used.

**Renvoie**

List of object uuids

**Lève**

- ValueError** – if panel is unknown

**get\_sel\_object\_uuids**(*include\_groups* : *bool* = *False*) → *list*[*str*]

Return selected objects uuids.

**Paramètres**

**include\_groups** – If *True*, also return objects from selected groups.

**Renvoie**

List of selected objects uuids.

**select\_objects**(*selection* : *list*[*int* | *str*], *panel* : *str* | *None* = *None*) → *None*

Select objects in current panel.

**Paramètres**

- **selection** – List of object numbers (1 to N) or uuids to select
- **panel** – panel name (valid values : « signal », « image »). If *None*, current panel is used. Defaults to *None*.

**select\_groups**(*selection* : *list*[*int* | *str*] | *None* = *None*, *panel* : *str* | *None* = *None*) → *None*

Select groups in current panel.

**Paramètres**

- **selection** – List of group numbers (1 to N), or list of group uuids, or *None* to select all groups. Defaults to *None*.
- **panel** – panel name (valid values : « signal », « image »). If *None*, current panel is used. Defaults to *None*.

**delete\_metadata**(*refresh\_plot* : *bool* = *True*, *keep\_roi* : *bool* = *False*) → *None*

Delete metadata of selected objects

**Paramètres**

- **refresh\_plot** – Refresh plot. Defaults to *True*.
- **keep\_roi** – Keep ROI. Defaults to *False*.

**get\_object\_shapes**(*nb\_id\_title* : *int* | *str* | *None* = *None*, *panel* : *str* | *None* = *None*) → *list*

Get plot item shapes associated to object (signal/image).

**Paramètres**

- **nb\_id\_title** – Object number, or object id, or object title. Defaults to *None* (current object).
- **panel** – Panel name. Defaults to *None* (current panel).

**Renvoie**

List of plot item shapes

**add\_annotations\_from\_items**(*items* : *list*, *refresh\_plot* : *bool* = *True*, *panel* : *str* | *None* = *None*) → *None*

Add object annotations (annotation plot items).

**Paramètres**

- **items** – annotation plot items
- **refresh\_plot** – refresh plot. Defaults to True.
- **panel** – panel name (valid values : « signal », « image »). If None, current panel is used.

**add\_label\_with\_title**(title : *str* | *None* = *None*, panel : *str* | *None* = *None*) → *None*

Add a label with object title on the associated plot

**Paramètres**

- **title** – Label title. Defaults to None. If None, the title is the object title.
- **panel** – panel name (valid values : « signal », « image »). If None, current panel is used.

**run\_macro**(number\_or\_title : *int* | *str* | *None* = *None*) → *None*

Run macro.

**Paramètres**

- number** – Number of the macro (starting at 1). Defaults to None (run current macro, or does nothing if there is no macro).

**stop\_macro**(number\_or\_title : *int* | *str* | *None* = *None*) → *None*

Stop macro.

**Paramètres**

- number** – Number of the macro (starting at 1). Defaults to None (stop current macro, or does nothing if there is no macro).

**import\_macro\_from\_file**(filename : *str*) → *None*

Import macro from file

**Paramètres**

- filename** – Filename.

**property panels**: **tuple**[*AbstractPanel*, ...]

Return the tuple of implemented panels (signal, image)

**Renvoie**

Tuple of panels

**confirm\_memory\_state**() → *bool*

Check memory warning state and eventually show a warning dialog

**Renvoie**

True if memory state is ok

**check\_stable\_release**() → *None*

Check if this is a stable release

**check\_for\_previous\_crash**() → *None*

Check for previous crash

**execute\_post\_show\_actions**() → *None*

Execute post-show actions

**take\_screenshot**(name : *str*) → *None*

Take main window screenshot

**take\_menu\_screenshots**() → *None*

Take menu screenshots

**setup**(console : *bool* = *False*) → *None*

Setup main window

**Paramètres**

- console** – True to setup console

**set\_process\_isolation\_enabled**(*state* : *bool*) → *None*

Enable/disable process isolation

**Paramètres**

**state** – True to enable process isolation

**get\_current\_panel**() → *str*

Return current panel name

**Renvoie**

« signal », « image », « macro »)

**Type renvoyé**

Panel name (valid values

**set\_current\_panel**(*panel* : *str*) → *None*

Switch to panel.

**Paramètres**

**panel** – panel name (valid values : « signal », « image », « macro »)

**Lève**

**ValueError** – unknown panel

**calc**(*name* : *str*, *param* : *DataSet* | *None* = *None*) → *None*

Call compute function name in current panel's processor.

**Paramètres**

— **name** – Compute function name

— **param** – Compute function parameter. Defaults to None.

**Lève**

**ValueError** – unknown function

**has\_objects**() → *bool*

Return True if sig/ima panels have any object

**set\_modified**(*state* : *bool* = *True*) → *None*

Set mainwindow modified state

**repopulate\_panel\_trees**() → *None*

Repopulate all panel trees

**toggle\_show\_titles**(*state* : *bool*) → *None*

Toggle show annotations option

**Paramètres**

**state** – state

**toggle\_auto\_refresh**(*state* : *bool*) → *None*

Toggle auto refresh option

**Paramètres**

**state** – state

**toggle\_show\_first\_only**(*state* : *bool*) → *None*

Toggle show first only option

**Paramètres**

**state** – state

**reset\_all**() → *None*

Reset all application data

**save\_to\_h5\_file**(filename=None) → None

Save to a DataLab HDF5 file

**Paramètres**

**filename** – HDF5 filename. If None, a file dialog is opened.

**Lève**

**IOError** – if filename is invalid or file cannot be saved.

**open\_h5\_files**(h5files : list[str] | None = None, import\_all : bool | None = None, reset\_all : bool | None = None) → None

Open a DataLab HDF5 file or import from any other HDF5 file.

**Paramètres**

— **h5files** – HDF5 filenames (optionally with dataset name, separated by « : »)

— **import\_all** – Import all datasets from HDF5 files

— **reset\_all** – Reset all application data before importing

**browse\_h5\_files**(filenames : list[str], reset\_all : bool) → None

Browse HDF5 files

**Paramètres**

— **filenames** – HDF5 filenames

— **reset\_all** – Reset all application data before importing

**import\_h5\_file**(filename : str, reset\_all : bool | None = None) → None

Import HDF5 file into DataLab

**Paramètres**

— **filename** – HDF5 filename (optionally with dataset name,

— " (separated by) – « »)

— **reset\_all** – Delete all DataLab signals/images before importing data

**add\_object**(obj : SignalObj | ImageObj) → None

Add object - signal or image

**Paramètres**

**obj** – object to add (signal or image)

**load\_from\_files**(filenames : list[str]) → None

Open objects from files in current panel (signals/images)

**Paramètres**

**filenames** – list of filenames

**get\_version**() → str

Return DataLab public version.

**Renvoie**

DataLab version

**close\_application**() → None

Close DataLab application

**raise\_window**() → None

Raise DataLab window

**add\_signal**(title : str, xdata : ndarray, ydata : ndarray, xunit : str | None = None, yunit : str | None = None, xlabel : str | None = None, ylabel : str | None = None) → bool

Add signal data to DataLab.

**Paramètres**

— **title** – Signal title

- **xdata** – X data
- **ydata** – Y data
- **xunit** – X unit. Defaults to None.
- **yunit** – Y unit. Defaults to None.
- **xlabel** – X label. Defaults to None.
- **ylabel** – Y label. Defaults to None.

**Renvoie**

True if signal was added successfully, False otherwise

**Lève**

- **ValueError** – Invalid xdata dtype
- **ValueError** – Invalid ydata dtype

**add\_image**(*tiile* : *str*, *data* : *ndarray*, *xunit* : *str* | *None* = *None*, *yunit* : *str* | *None* = *None*, *zunit* : *str* | *None* = *None*, *xlabel* : *str* | *None* = *None*, *ylabel* : *str* | *None* = *None*, *zlabel* : *str* | *None* = *None*) → bool

Add image data to DataLab.

**Paramètres**

- **title** – Image title
- **data** – Image data
- **xunit** – X unit. Defaults to None.
- **yunit** – Y unit. Defaults to None.
- **zunit** – Z unit. Defaults to None.
- **xlabel** – X label. Defaults to None.
- **ylabel** – Y label. Defaults to None.
- **zlabel** – Z label. Defaults to None.

**Renvoie**

True if image was added successfully, False otherwise

**Lève**

- **ValueError** – Invalid data dtype

**play\_demo**() → *None*

Play demo

**show\_tour**() → *None*

Show tour

**static test\_segfault\_error**() → *None*

Generate errors (both fault and traceback)

**show**() → *None*

Reimplement QMainWindow method

**close\_properly**() → bool

Close properly

**Renvoie**

True if closed properly, False otherwise

**closeEvent**(*event* : *QCloseEvent*) → *None*

Reimplement QMainWindow method

## 3.8 Panneau

Le paquet `cdl.core.gui.panel` fournit les **objets panneaux** pour les signaux et les images.

Trois types de panneaux sont disponibles :

- `cdl.core.gui.panel.signal.SignalPanel` : Panneau Signal
- `cdl.core.gui.panel.image.ImagePanel` : Panneau Image
- `cdl.core.gui.panel.macro.MacroPanel` : Panneau Macro

Les panneaux Signal et Image sont appelés **Panneaux de données** et sont utilisés pour afficher et gérer les signaux et les images dans la fenêtre principale de DataLab.

Les Panneaux de données reposent sur la classe `cdl.core.gui.panel.base.ObjectProp` (gestion des propriétés de l'objet) et un ensemble de modules pour gérer les fonctionnalités de l'interface graphique :

- `cdl.core.gui.actionhandler` : Actions de l'application (menus, barres d'outils, menu contextuel)
- `cdl.core.gui.objectview` : Widgets pour afficher les arbres d'objets (signaux/images)
- `cdl.core.gui.plothandler` : items *PlotPy* de représentation graphique de signaux et d'images
- `cdl.core.gui.processor` : Processeur (calcul)
- `cdl.core.gui.panel.roieditor` : Éditeur de ROI

Le Panneau Macro est utilisé pour afficher et exécuter des macros. Il repose sur le module `cdl.core.gui.macroeditor` pour gérer l'édition et l'exécution des macros.

### 3.8.1 Fonctionnalités de base

`cdl.core.gui.panel.base.is_plot_item_serializable(item : ShapeTypes) → bool`

Renvoie True si l'élément de tracé est sérialisable

**class** `cdl.core.gui.panel.base.ObjectProp(panel : BaseDataPanel, paramclass : SignalObj | ImageObj)`

Objet gérant les propriétés du panneau

**add\_button**(button)

Add additional button on bottom of properties panel

**set\_param\_label**(param : SignalObj | ImageObj)

Set computing parameters label

**update\_properties\_from**(param : SignalObj | ImageObj | None = None)

Update properties from signal/image dataset

**class** `cdl.core.gui.panel.base.AbstractPanelMeta(name, bases, namespace, /, **kwargs)`

Mixed metaclass to avoid conflicts

**class** `cdl.core.gui.panel.base.AbstractPanel(parent)`

Object defining DataLab panel interface, based on a vertical QSplitter widget

A panel handle an object list (objects are signals, images, macros, ...). Each object must implement `cdl.core.gui.ObjItf` interface

**get\_serializable\_name**(obj : ObjItf) → str

Return serializable name of object

**serialize\_object\_to\_hdf5**(obj : ObjItf, writer : NativeH5Writer) → None

Serialize object to HDF5 file

**deserialize\_object\_from\_hdf5**(reader : NativeH5Reader, name : str) → ObjItf

Deserialize object from a HDF5 file

**abstract serialize\_to\_hdf5**(writer : NativeH5Writer) → None

Serialize whole panel to a HDF5 file

**abstract deserialize\_from\_hdf5**(reader : NativeH5Reader) → None

Deserialize whole panel from a HDF5 file

**abstract create\_object**() → ObjItf

Create and return object

**abstract add\_object**(obj : ObjItf) → None

Add object to panel

**abstract remove\_all\_objects**()

Remove all objects

**class** cdl.core.gui.panel.base.**ResultData**(results : list[ResultShape | ResultProperties] = None, xlabel : list[str] = None, ylabel : list[str] = None)

Result data associated to a shapetype

cdl.core.gui.panel.base.**create\_resultdata\_dict**(objs : list[SignalObj | ImageObj]) → dict[str, ResultData]

Return result data dictionary

**Paramètres**

**objs** – List of objects

**Renvoie**

keys are result categories, values are ResultData

**Type renvoyé**

Result data dictionary

**class** cdl.core.gui.panel.base.**PasteMetadataParam**

Paste metadata parameters

**keep\_roi**

Par défaut : True.

**Type**

guidata.dataset.dataitems.BoolItem

**keep\_resultshapes**

Par défaut : False.

**Type**

guidata.dataset.dataitems.BoolItem

**keep\_annotations**

Par défaut : True.

**Type**

guidata.dataset.dataitems.BoolItem

**keep\_resultproperties**

Par défaut : False.

**Type**

guidata.dataset.dataitems.BoolItem

**keep\_other**

Par défaut : True.

**Type**`guidata.dataset.dataitems.BoolItem`

**classmethod create**(*keep\_roi* : *bool*, *keep\_resultshapes* : *bool*, *keep\_annotations* : *bool*,  
*keep\_resultproperties* : *bool*, *keep\_other* : *bool*) →  
*cdl.core.gui.panel.base.PasteMetadataParam*

Renvoie une nouvelle instance de *PasteMetadataParam* avec les champs initialisés aux valeurs données.

**Paramètres**

- **keep\_roi** (*bool*) – Par défaut : True.
- **keep\_resultshapes** (*bool*) – Par défaut : False.
- **keep\_annotations** (*bool*) – Par défaut : True.
- **keep\_resultproperties** (*bool*) – Par défaut : False.
- **keep\_other** (*bool*) – Par défaut : True.

**Renvoie**

Nouvelle instance de *PasteMetadataParam*.

**class** `cdl.core.gui.panel.base.BaseDataPanel`(*parent* : *QWidget*)

Object handling the item list, the selected item properties and plot

**abstract static get\_roieditor\_class**() → *Type*[*TypeROIEditor*]

Return ROI editor class

**closeEvent**(*event*)

Reimplement QMainWindow method

**plot\_item\_parameters\_changed**(*item* : *CurveItem* | *MaskedImageItem* | *LabelItem*) → *None*

Plot items changed : update metadata of all objects from plot items

**plot\_item\_moved**(*item* : *LabelItem*, *x0* : *float*, *y0* : *float*, *x1* : *float*, *y1* : *float*) → *None*

Plot item moved : update metadata of all objects from plot items

**Paramètres**

- **item** – Plot item
- **x0** – new x0 coordinate
- **y0** – new y0 coordinate
- **x1** – new x1 coordinate
- **y1** – new y1 coordinate

**serialize\_object\_to\_hdf5**(*obj* : *TypeObj*, *writer* : *NativeH5Writer*) → *None*

Serialize object to HDF5 file

**serialize\_to\_hdf5**(*writer* : *NativeH5Writer*) → *None*

Serialize whole panel to a HDF5 file

**deserialize\_from\_hdf5**(*reader* : *NativeH5Reader*) → *None*

Deserialize whole panel from a HDF5 file

**create\_object**() → *TypeObj*

Create object (signal or image)

**Renvoie**

SignalObj or ImageObj object

**add\_object**(*obj* : *TypeObj*, *group\_id* : *str* | *None* = *None*, *set\_current* : *bool* = *True*) → *None*

Add object

**Paramètres**

- **obj** – SignalObj or ImageObj object
- **group\_id** – group id

— **set\_current** – if True, set the added object as current

**remove\_all\_objects()** → None

Remove all objects

**setup\_panel()** → None

Setup panel

**refresh\_plot**(*what* : str, *update\_items* : bool = True, *force* : bool = False) → None

Refresh plot. This method simply emits the signal SIG\_REFRESH\_PLOT which is connected to the method *PlotHandler.refresh\_plot*.

**Paramètres**

- **what** – string describing the objects to refresh. Valid values are « selected » (refresh the selected objects), « all » (refresh all objects), « existing » (refresh existing plot items), or an object uuid.
- **update\_items** – if True, update the items. If False, only show the items (do not update them, except if the option « Use reference item LUT range » is enabled and more than one item is selected). Defaults to True.
- **force** – if True, force refresh even if auto refresh is disabled, and refresh all items associated to objects (even the hidden ones, e.g. when selecting multiple images of the same size and position). Defaults to False.

**Lève**

**ValueError** – if *what* is not a valid value

**manual\_refresh()** → None

Manual refresh

**get\_category\_actions**(*category* : ActionCategory) → list[QAction]

Return actions for category

**get\_context\_menu()** → QMenu

Update and return context menu

**add\_group**(*title* : str) → ObjectGroup

Add group

**duplicate\_object()** → None

Duplication signal/image object

**copy\_metadata()** → None

Copy object metadata

**paste\_metadata**(*param* : PasteMetadataParam | None = None) → None

Paste metadata to selected object(s)

**remove\_object**(*force* : bool = False) → None

Remove signal/image object

**Paramètres**

**force** – if True, remove object without confirmation. Defaults to False.

**delete\_all\_objects()** → None

Confirm before removing all objects

**delete\_metadata**(*refresh\_plot* : bool = True, *keep\_roi* : bool | None = None) → None

Delete metadata of selected objects

**Paramètres**

— **refresh\_plot** – Refresh plot. Defaults to True.

— **keep\_roi** – Keep regions of interest, if any. Defaults to None (ask user).

**add\_annotations\_from\_items**(*items* : *list*, *refresh\_plot* : *bool* = *True*) → *None*

Add object annotations (annotation plot items).

**Paramètres**

— **items** – annotation plot items  
— **refresh\_plot** – refresh plot. Defaults to True.

**update\_metadata\_view\_settings**() → *None*

Update metadata view settings

**copy\_titles\_to\_clipboard**() → *None*

Copy object titles to clipboard (for reproducibility)

**new\_group**() → *None*

Create a new group

**rename\_group**(*new\_name* : *str* | *None* = *None*) → *None*

Rename a group

**Paramètres**

**new\_name** – new group name. Defaults to None (ask user).

**abstract get\_newparam\_from\_current**(*newparam* : *NewSignalParam* | *NewImageParam* | *None* = *None*) → *NewSignalParam* | *NewImageParam* | *None*

Get new object parameters from the current object.

**Paramètres**

**newparam** – new object parameters. If None, create a new one.

**Renvoie**

New object parameters

**abstract new\_object**(*newparam* : *NewSignalParam* | *NewImageParam* | *None* = *None*, *addparam* : *gds.DataSet* | *None* = *None*, *edit* : *bool* = *True*, *add\_to\_panel* : *bool* = *True*) → *TypeObj* | *None*

Create a new object (signal/image).

**Paramètres**

— **newparam** – new object parameters  
— **addparam** – additional parameters  
— **edit** – Open a dialog box to edit parameters (default : True)  
— **add\_to\_panel** – Add object to panel (default : True)

**Renvoie**

New object

**set\_current\_object\_title**(*title* : *str*) → *None*

Set current object title

**load\_from\_files**(*filenames* : *list[str]* | *None* = *None*) → *list[TypeObj]*

Open objects from file (signals/images), add them to DataLab and return them.

**Paramètres**

**filenames** – File names

**Renvoie**

list of new objects

**save\_to\_files**(*filenames* : *list[str]* | *str* | *None* = *None*) → *None*

Save selected objects to files (signal/image).

**Paramètres****filenames** – File names**handle\_dropped\_files**(*filenames* : *list[str]* | *None* = *None*) → *None*

Handle dropped files

**Paramètres****filenames** – File names**Renvoie***None***exec\_import\_wizard**() → *None*

Execute import wizard

**import\_metadata\_from\_file**(*filename* : *str* | *None* = *None*) → *None*

Import metadata from file (JSON).

**Paramètres****filename** – File name**export\_metadata\_from\_file**(*filename* : *str* | *None* = *None*) → *None*

Export metadata to file (JSON).

**Paramètres****filename** – File name**selection\_changed**(*update\_items* : *bool* = *False*) → *None*

Object selection changed : update object properties, refresh plot and update object view.

**Paramètres****update\_items** – Update plot items (default : *False*)**properties\_changed**() → *None*

The properties “Apply” button was clicked : update object properties, refresh plot and update object view.

**add\_plot\_items\_to\_dialog**(*dlg* : *PlotDialog*, *oids* : *list[str]*) → *None*

Add plot items to dialog

**Paramètres**— **dlg** – Dialog— **oids** – Object IDs**open\_separate\_view**(*oids* : *list[str]* | *None* = *None*, *edit\_annotations* : *bool* = *False*) → *PlotDialog* | *None*

Open separate view for visualizing selected objects

**Paramètres**— **oids** – Object IDs (default : *None*)— **edit\_annotations** – Edit annotations (default : *False*)**Renvoie**Instance of *PlotDialog***create\_new\_dialog**(*edit* : *bool* = *False*, *toolbar* : *bool* = *True*, *title* : *str* | *None* = *None*, *name* : *str* | *None* = *None*, *options* : *dict* | *None* = *None*) → *PlotDialog* | *None*

Create new pop-up signal/image plot dialog.

**Paramètres**— **edit** – Edit mode— **toolbar** – Show toolbar— **title** – Dialog title— **name** – Dialog object name— **options** – Plot options

**Renvoie**

Plot dialog instance

**get\_roi\_editor\_output**(*extract* : *bool*) → tuple[TypeROI, bool] | None

Get ROI data (array) from specific dialog box.

**Paramètres****extract** – Extract ROI from data**Renvoie**

A tuple containing the ROI object and a boolean indicating whether the dialog was accepted or not.

**get\_objects\_with\_dialog**(*title* : *str*, *comment* : *str* = "", *nb\_objects* : *int* = 1, *parent* : *QWidget* | *None* = *None*) → TypeObj | None

Get object with dialog box.

**Paramètres**

- **title** – Dialog title
- **comment** – Optional dialog comment
- **nb\_objects** – Number of objects to select
- **parent** – Parent widget

**Renvoie**

Object(s) (signal(s) or image(s), or None if dialog was canceled)

**add\_objprop\_buttons**() → None

Insert additional buttons in object properties panel

**show\_results**() → None

Show results

**plot\_results**() → None

Plot results

**delete\_results**() → None

Delete results

**add\_label\_with\_title**(*title* : *str* | *None* = *None*) → None

Add a label with object title on the associated plot

**Paramètres****title** – Label title. Defaults to None. If None, the title is the object title.

### 3.8.2 Signal panel

```
class cdl.core.gui.panel.signal.SignalPanel(parent : QW.QWidget, dockableplotwidget :
                                           DockablePlotWidget, panel_toolbar : QW.QToolBar)
```

Object handling the item list, the selected item properties and plot, specialized for Signal objects

**PARAMCLASS**

Signal object

**uuid**

Par défaut : None.

**Type**

guidata.dataset.dataitems.StringItem

**xydata**

Données. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatArrayItem`

**metadata**

Métadonnées. Par défaut : {}.

**Type**

`guidata.dataset.dataitems.DictItem`

**title**

Titre du signal. Par défaut : “Sans titre”.

**Type**

`guidata.dataset.dataitems.StringItem`

**xlabel**

Titre. Par défaut : “”.

**Type**

`guidata.dataset.dataitems.StringItem`

**xunit**

Unité. Par défaut : “”.

**Type**

`guidata.dataset.dataitems.StringItem`

**ylabel**

Titre. Par défaut : “”.

**Type**

`guidata.dataset.dataitems.StringItem`

**yunit**

Unité. Par défaut : “”.

**Type**

`guidata.dataset.dataitems.StringItem`

**autoscale**

Par défaut : True.

**Type**

`guidata.dataset.dataitems.BoolItem`

**xscalelog**

Par défaut : False.

**Type**

`guidata.dataset.dataitems.BoolItem`

**xscalemin**

Borne inférieure. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**xscalemax**

Borne supérieure. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**yscalelog**

Par défaut : False.

**Type**

`guidata.dataset.dataitems.BoolItem`

**yscalemin**

Borne inférieure. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**yscalemax**

Borne supérieure. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

alias de *SignalObj*

**classmethod** `PARAMCLASS.create(uuid : str, xydata : numpy.ndarray, metadata : dict, title : str, xlabel : str, xunit : str, ylabel : str, yunit : str, autoscale : bool, xscalelog : bool, xscalemin : float, xscalemax : float, yscalelog : bool, yscalemin : float, yscalemax : float) → cdl.core.model.signal.SignalObj`

Renvoie une nouvelle instance de *SignalObj* avec les champs initialisés aux valeurs données.

**Paramètres**

- **uuid** (*str*) – Par défaut : None.
- **xydata** (*numpy.ndarray*) – Données. Par défaut : None.
- **metadata** (*dict*) – Métadonnées. Par défaut : {}.
- **title** (*str*) – Titre du signal. Par défaut : “Sans titre”.
- **xlabel** (*str*) – Titre. Par défaut : “”.
- **xunit** (*str*) – Unité. Par défaut : “”.
- **ylabel** (*str*) – Titre. Par défaut : “”.
- **yunit** (*str*) – Unité. Par défaut : “”.
- **autoscale** (*bool*) – Par défaut : True.
- **xscalelog** (*bool*) – Par défaut : False.
- **xscalemin** (*float*) – Borne inférieure. Par défaut : None.
- **xscalemax** (*float*) – Borne supérieure. Par défaut : None.
- **yscalelog** (*bool*) – Par défaut : False.
- **yscalemin** (*float*) – Borne inférieure. Par défaut : None.
- **yscalemax** (*float*) – Borne supérieure. Par défaut : None.

**Renvoie**

Nouvelle instance de *SignalObj*.

**IO\_REGISTRY**

alias de *SignalIORegistry*

**static** `get_roieditor_class() → Type[SignalROIEditor]`

Return ROI editor class

**get\_newparam\_from\_current** (*newparam : NewSignalParam | None = None, title : str | None = None*) → *NewSignalParam | None*

Get new object parameters from the current object.

**Paramètres**

- **newparam** (*guidata.dataset.DataSet*) – new object parameters. If None, create a new one.
- **title** – new object title. If None, use the current object title, or the default title.

**Renvoie**

New object parameters

**new\_object**(*newparam* : [NewSignalParam](#) | *None* = *None*, *addparam* : *gds.DataSet* | *None* = *None*, *edit* : *bool* = *True*, *add\_to\_panel* : *bool* = *True*) → [SignalObj](#) | *None*

Create a new object (signal).

**Paramètres**

- **newparam** ([guidata.dataset.DataSet](#)) – new object parameters
- **addparam** ([guidata.dataset.DataSet](#)) – additional parameters
- **edit** (*bool*) – Open a dialog box to edit parameters (default : *True*)
- **add\_to\_panel** (*bool*) – Add the new object to the panel (default : *True*)

**Renvoie**

New object

**toggle\_anti\_aliasing**(*state* : *bool*) → *None*

Toggle anti-aliasing on/off

**Paramètres**

**state** – state of the anti-aliasing

**reset\_curve\_styles**() → *None*

Reset curve styles

### 3.8.3 Image panel

**class** `cdl.core.gui.panel.image.ImagePanel`(*parent* : *QW.QWidget*, *dockableplotwidget* : [DockablePlotWidget](#), *panel\_toolbar* : *QW.QToolBar*)

Object handling the item list, the selected item properties and plot, specialized for Image objects

**PARAMCLASS**

Image object

**uuid**Par défaut : *None*.**Type**[guidata.dataset.dataitems.StringItem](#)**data**Données. Par défaut : *None*.**Type**[guidata.dataset.dataitems.FloatArrayItem](#)**metadata**Métadonnées. Par défaut : *{ }*.**Type**[guidata.dataset.dataitems.DictItem](#)**x0**X<sub>0</sub>. Par défaut : 0.0.**Type**[guidata.dataset.dataitems.FloatItem](#)**y0**Y<sub>0</sub>. Par défaut : 0.0.

**Type**

guidata.dataset.dataitems.FloatItem

**dx**

x. Flottant, non nul. Par défaut : 1.0.

**Type**

guidata.dataset.dataitems.FloatItem

**dy**

y. Flottant, non nul. Par défaut : 1.0.

**Type**

guidata.dataset.dataitems.FloatItem

**title**

Titre de l'image. Par défaut : "Sans titre".

**Type**

guidata.dataset.dataitems.StringItem

**xlabel**

Titre. Par défaut : "".

**Type**

guidata.dataset.dataitems.StringItem

**xunit**

Unité. Par défaut : "".

**Type**

guidata.dataset.dataitems.StringItem

**ylabel**

Titre. Par défaut : "".

**Type**

guidata.dataset.dataitems.StringItem

**yunit**

Unité. Par défaut : "".

**Type**

guidata.dataset.dataitems.StringItem

**zlabel**

Titre. Par défaut : "".

**Type**

guidata.dataset.dataitems.StringItem

**zunit**

Unité. Par défaut : "".

**Type**

guidata.dataset.dataitems.StringItem

**autoscale**

Par défaut : True.

**Type**

guidata.dataset.dataitems.BoolItem

**xscalelog**

Par défaut : False.

**Type**

`guidata.dataset.dataitems.BoolItem`

**xscalemin**

Borne inférieure. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**xscalemax**

Borne supérieure. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**yscalelog**

Par défaut : False.

**Type**

`guidata.dataset.dataitems.BoolItem`

**yscalemin**

Borne inférieure. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**yscalemax**

Borne supérieure. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**zscalemin**

Borne inférieure. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

**zscalemax**

Borne supérieure. Par défaut : None.

**Type**

`guidata.dataset.dataitems.FloatItem`

alias de `ImageObj`

```
classmethod PARAMCLASS.create(uuid : str, data : numpy.ndarray, metadata : dict, x0 : float, y0 : float,  
                                dx : float, dy : float, title : str, xlabel : str, xunit : str, ylabel : str, yunit :  
                                str, zlabel : str, zunit : str, autoscale : bool, xscalelog : bool,  
                                xscalemin : float, xscalemax : float, yscalelog : bool, yscalemin : float,  
                                yscalemax : float, zscalemin : float, zscalemax : float) →  
                                cdl.core.model.image.ImageObj
```

Renvoie une nouvelle instance de `ImageObj` avec les champs initialisés aux valeurs données.

**Paramètres**

- **uuid** (*str*) – Par défaut : None.
- **data** (*numpy.ndarray*) – Données. Par défaut : None.
- **metadata** (*dict*) – Métadonnées. Par défaut : {}.
- **x0** (*float*) –  $X_0$ . Par défaut : 0.0.

- **y0** (*float*) –  $Y_0$ . Par défaut : 0.0.
- **dx** (*float*) – x. Flottant, non nul. Par défaut : 1.0.
- **dy** (*float*) – y. Flottant, non nul. Par défaut : 1.0.
- **title** (*str*) – Titre de l'image. Par défaut : "Sans titre".
- **xlabel** (*str*) – Titre. Par défaut : "".
- **xunit** (*str*) – Unité. Par défaut : "".
- **ylabel** (*str*) – Titre. Par défaut : "".
- **yunit** (*str*) – Unité. Par défaut : "".
- **zlabel** (*str*) – Titre. Par défaut : "".
- **zunit** (*str*) – Unité. Par défaut : "".
- **autoscale** (*bool*) – Par défaut : True.
- **xscalelog** (*bool*) – Par défaut : False.
- **xscalemin** (*float*) – Borne inférieure. Par défaut : None.
- **xscalemax** (*float*) – Borne supérieure. Par défaut : None.
- **yscalelog** (*bool*) – Par défaut : False.
- **yscalemin** (*float*) – Borne inférieure. Par défaut : None.
- **yscalemax** (*float*) – Borne supérieure. Par défaut : None.
- **zscalemin** (*float*) – Borne inférieure. Par défaut : None.
- **zscalemax** (*float*) – Borne supérieure. Par défaut : None.

**Renvoie**

Nouvelle instance de ImageObj.

**IO\_REGISTRY**

alias de ImageIORegistry

**static get\_roieditor\_class()** → *Type[ImageROIEditor]*

Return ROI editor class

**plot\_lut\_changed**(*plot : BasePlot*) → *None*

The LUT of the plot has changed : updating image objects accordingly

**Paramètres**

**plot** – Plot object

**get\_newparam\_from\_current**(*newparam : NewImageParam | None = None, title : str | None = None*) → *NewImageParam | None*

Get new object parameters from the current object.

**Paramètres**

- **newparam** (*guidata.dataset.DataSet*) – new object parameters. If None, create a new one.
- **title** – new object title. If None, use the current object title, or the default title.

**Renvoie**

New object parameters

**new\_object**(*newparam : NewImageParam | None = None, addparam : gds.DataSet | None = None, edit : bool = True, add\_to\_panel : bool = True*) → *ImageObj | None*

Create a new object (image).

**Paramètres**

- **newparam** (*Daguidata.dataset.datatypes.DataSettaSet*) – new object parameters
- **addparam** (*guidata.dataset.DataSet*) – additional parameters
- **edit** (*bool*) – Open a dialog box to edit parameters (default : True)
- **add\_to\_panel** (*bool*) – Add the object to the panel (default : True)

**Renvoie**

New object

**toggle\_show\_contrast**(*state : bool*) → *None*

Toggle show contrast option

### 3.8.4 Macro panel

**class** `cdl.core.gui.panel.macro.MacroTabs`(*parent=None*)

Macro tabwidget

**Paramètres**

**parent** (*QWidget*) – Parent widget

**clear**() → *None*

Override Qt method

**contextMenuEvent**(*event*)

Override Qt method

**add\_tab**(*macro : Macro*) → *int*

Add tab

**Paramètres**

**macro** – Macro object

**Renvoie**

Number of the tab (starting at 1)

**Type renvoyé**

*int*

**remove\_tab**(*number : int*) → *None*

Remove tab

**Paramètres**

**number** – Number of the tab (starting at 1)

**get\_widget**(*number : int*) → *CodeEditor*

Return macro editor widget at number

**Paramètres**

**number** – Number of the tab (starting at 1)

**Renvoie**

Macro editor widget

**set\_current\_number**(*number : int*) → *None*

Set current tab number

**Paramètres**

**number** – Number of the tab (starting at 1)

**get\_current\_number**() → *int*

Return current tab number

**Renvoie**

Number of the tab (starting at 1)

**Type renvoyé**

*int*

**set\_tab\_title**(*number : int, name : str*) → *None*

Set tab title

**Paramètres**

- **number** – Number of the tab (starting at 1)
- **name** – Macro name

```
class cdl.core.gui.panel.macro.MacroPanel(parent : QWidget | None = None)
    Macro Panel widget

    Paramètres
    parent (QWidget) – Parent widget

    update_color_mode() → None
        Update color mode according to the current theme

    get_serializable_name(obj : Macro) → str
        Return serializable name of object

    serialize_to_hdf5(writer : NativeH5Writer) → None
        Serialize whole panel to a HDF5 file

    Paramètres
    writer – HDF5 writer

    deserialize_from_hdf5(reader : NativeH5Reader) → None
        Deserialize whole panel from a HDF5 file

    Paramètres
    reader – HDF5 reader

    create_object() → Macro
        Create object.

    Renvoi
    Macro object

    add_object(obj : Macro) → None
        Add object.

    Paramètres
    obj – Macro object

    remove_all_objects() → None
        Remove all objects

    setup_actions() → None
        Setup macro menu actions

    get_macro(number_or_title : int | str | None = None) → Macro | None
        Return macro at number (if number is None, return current macro)

    Paramètres
    number – Number of the macro (starting at 1) or title of the macro. Defaults to None (current macro).

    Renvoi
    Macro object or None (if not found)

    get_number_from_title(title : str) → int | None
        Return macro number from title

    Paramètres
    title – Title of the macro

    Renvoi
    Number of the macro (starting at 1) or None (if not found)
```

**get\_number\_from\_macro**(*macro* : *Macro*) → int | None

Return macro number from macro object

**Paramètres**

**macro** – Macro object

**Renvoie**

Number of the macro (starting at 1) or None (if not found)

**get\_macro\_titles**() → list[str]

Return list of macro titles

**macro\_contents\_changed**() → None

One of the macro contents has changed

**run\_macro**(*number\_or\_title* : int | str | None = None) → None

Run current macro

**Paramètres**

**number** – Number of the macro (starting at 1). Defaults to None (run current macro, or does nothing if there is no macro).

**stop\_macro**(*number\_or\_title* : int | str | None = None) → None

Stop current macro

**Paramètres**

**number** – Number of the macro (starting at 1). Defaults to None (run current macro, or does nothing if there is no macro).

**macro\_state\_changed**(*orig\_macro* : *Macro*, *state* : bool) → None

Macro state has changed (True : started, False : stopped)

**Paramètres**

— **orig\_macro** – Macro object

— **state** – State of the macro

**add\_macro**() → Macro

Add macro, optionally with name

**Renvoie**

Macro object

**macro\_name\_changed**(*name* : str) → None

Macro name has been changed

**Paramètres**

**name** – New name of the macro

**rename\_macro**(*number* : int | None = None, *title* : str | None = None) → None

Rename macro

**Paramètres**

— **number** – Number of the macro (starting at 1). Defaults to None.

— **title** – Title of the macro. Defaults to None.

**export\_macro\_to\_file**(*number\_or\_title* : int | str | None = None, *filename* : str | None = None) → None

Export macro to file

**Paramètres**

— **number\_or\_title** – Number of the macro (starting at 1) or title of the macro. Defaults to None.

— **filename** – Filename. Defaults to None.

Lève

**ValueError** – If title is not found

**import\_macro\_from\_file**(filename : *str* | *None* = *None*) → *int*

Import macro from file

**Paramètres**

**filename** – Filename. Defaults to None.

**Renvoie**

Number of the macro (starting at 1)

**remove\_macro**(number\_or\_title : *int* | *str* | *None* = *None*) → *None*

Remove macro

**Paramètres**

**number\_or\_title** – Number of the macro (starting at 1) or title of the macro. Defaults to None.

## 3.9 Gestionnaire d’actions

Le module `cdl.core.gui.actionhandler` gère toutes les actions de l’application (menus, barres d’outils, menu contextuel). Ces actions pointent vers les panneaux, les processeurs, les gestionnaires d’objets, ...

### 3.9.1 Classes utilitaires

**class** `cdl.core.gui.actionhandler.SelectCond`

Conditions de sélection de signaux ou d’images

**static** **always**(selected\_groups : *list*[*ObjectGroup*], selected\_objects : *list*[*SignalObj* | *ImageObj*]) → *bool*

Toujours vrai

**static** **exactly\_one**(selected\_groups : *list*[*ObjectGroup*], selected\_objects : *list*[*SignalObj* | *ImageObj*]) → *bool*

Exactement un signal ou une image est sélectionné

**static** **exactly\_one\_group**(selected\_groups : *list*[*ObjectGroup*], selected\_objects : *list*[*SignalObj* | *ImageObj*]) → *bool*

Exactement un groupe est sélectionné

**static** **at\_least\_one\_group\_or\_one\_object**(sel\_groups : *list*[*ObjectGroup*], sel\_objects : *list*[*SignalObj* | *ImageObj*]) → *bool*

Au moins un groupe ou un signal ou une image est sélectionné

**static** **at\_least\_one**(sel\_groups : *list*[*ObjectGroup*], sel\_objects : *list*[*SignalObj* | *ImageObj*]) → *bool*

Au moins un signal ou une image est sélectionné

**static** **at\_least\_two**(sel\_groups : *list*[*ObjectGroup*], sel\_objects : *list*[*SignalObj* | *ImageObj*]) → *bool*

Au moins deux signaux ou images sont sélectionnés

**static** **with\_roi**(selected\_groups : *list*[*ObjectGroup*], selected\_objects : *list*[*SignalObj* | *ImageObj*]) → *bool*

Au moins un signal ou une image a une ROI

```
class cdl.core.gui.actionhandler.ActionCategory(value, names=None, *, module=None,
                                              qualname=None, type=None, start=1,
                                              boundary=None)
```

Catégories d'actions

### 3.9.2 Classes de gestion

```
class cdl.core.gui.actionhandler.SignalActionHandler(panel : SignalPanel | ImagePanel,
                                                    panel_toolbar : QW.QToolBar, view_toolbar :
                                                    QW.QToolBar)
```

Objet gérant les interactions GUI du panneau de signaux : actions, menus, ...

**create\_first\_actions()**

Créer des actions qui sont ajoutées aux menus en premier lieu

**create\_last\_actions()**

Créer des actions qui sont ajoutées aux menus à la fin

**add\_action**(action : QAction, select\_condition : Callable | None = None) → None

Ajouter une action à la liste des actions.

**Paramètres**

- **action** – action à ajouter
- **select\_condition** – condition pour activer l'action. Par défaut à None. Si None, l'action est activée si au moins un objet est sélectionné.

**add\_to\_action\_list**(action : QAction, category : ActionCategory | None = None, pos : int | None = None, sep : bool = False) → None

Ajouter une action à la liste des actions.

**Paramètres**

- **action** – action à ajouter
- **category** – catégorie d'action. Par défaut à None. Si None, l'action est ajoutée à la catégorie actuelle.
- **pos** – ajouter l'action au menu à cette position. Par défaut à None. Si None, l'action est ajoutée à la fin de la liste.
- **sep** – ajouter un séparateur avant l'action dans le menu (ou après si pos est positif). Par défaut à False.

**create\_all\_actions()**

Créer toutes les actions

**new\_action**(title : str, position : int | None = None, separator : bool = False, triggered : Callable | None = None, toggled : Callable | None = None, shortcut : QShortcut | None = None, icon\_name : str | None = None, tip : str | None = None, select\_condition : Callable | str | None = None, context\_menu\_pos : int | None = None, context\_menu\_sep : bool = False, toolbar\_pos : int | None = None, toolbar\_sep : bool = False, toolbar\_category : ActionCategory | None = None) → QAction

Créer une nouvelle action et l'ajouter à la liste des actions.

**Paramètres**

- **title** – titre de l'action
- **position** – ajouter l'action au menu à cette position. Par défaut à None.
- **separator** – ajouter un séparateur avant l'action dans le menu (ou après si pos est positif). Par défaut à False.
- **triggered** – callback déclenché. Par défaut à None.

- **toggled** – callback de type toggle. Par défaut à None.
- **shortcut** – raccourci. Par défaut à None.
- **icon\_name** – nom de l’icône. Par défaut à None.
- **tip** – info-bulle. Par défaut à None.
- **select\_condition** – condition de sélection. Par défaut à None. Si str, doit être le nom d’une méthode de SelectCond, c’est-à-dire l’une des valeurs suivantes : « always », « exactly\_one », « exactly\_one\_group », « at\_least\_one\_group\_or\_one\_object », « at\_least\_one », « at\_least\_two », « with\_roi ».
- **context\_menu\_pos** – ajouter l’action au menu contextuel à cette position. Par défaut à None.
- **context\_menu\_sep** – ajouter un séparateur avant l’action dans le menu contextuel (ou après si context\_menu\_pos est positif). Par défaut à False.
- **toolbar\_pos** – ajouter l’action à la barre d’outils à cette position. Par défaut à None.
- **toolbar\_sep** – ajouter un séparateur avant l’action dans la barre d’outils (ou après si toolbar\_pos est positif). Par défaut à False.
- **toolbar\_category** – catégorie de la barre d’outils. Par défaut à None. S’il n’est pas None, cela spécifie la catégorie de la barre d’outils. Si None, par défaut à ActionCategory.VIEW\_TOOLBAR si la catégorie actuelle est ActionCategory.VIEW, sinon à ActionCategory.PANEL\_TOOLBAR.

**Renvoie**

Nouvelle action

**new\_category**(category : [ActionCategory](#)) → [Generator](#)[None, None, None]

Gestionnaire de contexte pour créer un nouveau menu.

**Paramètres****category** – Catégorie d’action**Renvoie (Yields)**

None

**new\_menu**(title : str, icon\_name : str | None = None) → [Generator](#)[None, None, None]

Gestionnaire de contexte pour créer un nouveau menu.

**Paramètres**— **title** – Titre du menu— **icon\_name** – Nom de l’icône du menu. Par défaut à None.**Renvoie (Yields)**

None

**selected\_objects\_changed**(selected\_groups : list[ObjectGroup], selected\_objects : list[SignalObj | ImageObj]) → None

Mettre à jour les actions en fonction des objets sélectionnés.

**Paramètres**— **selected\_groups** – groupes sélectionnés— **selected\_objects** – objets sélectionnés

```
class cdl.core.gui.actionhandler.ImageActionHandler(panel : SignalPanel | ImagePanel,
                                                    panel_toolbar : QW.QToolBar, view_toolbar :
                                                    QW.QToolBar)
```

Objet gérant les interactions GUI du panneau d’images : actions, menus, ...

**create\_first\_actions**()

Créer des actions qui sont ajoutées aux menus en premier lieu

**create\_last\_actions**()

Créer des actions qui sont ajoutées aux menus à la fin

**add\_action**(*action* : QAction, *select\_condition* : Callable | None = None) → None

Ajouter une action à la liste des actions.

**Paramètres**

- **action** – action à ajouter
- **select\_condition** – condition pour activer l’action. Par défaut à None. Si None, l’action est activée si au moins un objet est sélectionné.

**add\_to\_action\_list**(*action* : QAction, *category* : ActionCategory | None = None, *pos* : int | None = None, *sep* : bool = False) → None

Ajouter une action à la liste des actions.

**Paramètres**

- **action** – action à ajouter
- **category** – catégorie d’action. Par défaut à None. Si None, l’action est ajoutée à la catégorie actuelle.
- **pos** – ajouter l’action au menu à cette position. Par défaut à None. Si None, l’action est ajoutée à la fin de la liste.
- **sep** – ajouter un séparateur avant l’action dans le menu (ou après si pos est positif). Par défaut à False.

**create\_all\_actions**()

Créer toutes les actions

**new\_action**(*title* : str, *position* : int | None = None, *separator* : bool = False, *triggered* : Callable | None = None, *toggled* : Callable | None = None, *shortcut* : QShortcut | None = None, *icon\_name* : str | None = None, *tip* : str | None = None, *select\_condition* : Callable | str | None = None, *context\_menu\_pos* : int | None = None, *context\_menu\_sep* : bool = False, *toolbar\_pos* : int | None = None, *toolbar\_sep* : bool = False, *toolbar\_category* : ActionCategory | None = None) → QAction

Créer une nouvelle action et l’ajouter à la liste des actions.

**Paramètres**

- **title** – titre de l’action
- **position** – ajouter l’action au menu à cette position. Par défaut à None.
- **separator** – ajouter un séparateur avant l’action dans le menu (ou après si pos est positif). Par défaut à False.
- **triggered** – callback déclenché. Par défaut à None.
- **toggled** – callback de type toggle. Par défaut à None.
- **shortcut** – raccourci. Par défaut à None.
- **icon\_name** – nom de l’icône. Par défaut à None.
- **tip** – info-bulle. Par défaut à None.
- **select\_condition** – condition de sélection. Par défaut à None. Si str, doit être le nom d’une méthode de SelectCond, c’est-à-dire l’une des valeurs suivantes : « always », « exactly\_one », « exactly\_one\_group », « at\_least\_one\_group\_or\_one\_object », « at\_least\_one », « at\_least\_two », « with\_roi ».
- **context\_menu\_pos** – ajouter l’action au menu contextuel à cette position. Par défaut à None.
- **context\_menu\_sep** – ajouter un séparateur avant l’action dans le menu contextuel (ou après si context\_menu\_pos est positif). Par défaut à False.
- **toolbar\_pos** – ajouter l’action à la barre d’outils à cette position. Par défaut à None.
- **toolbar\_sep** – ajouter un séparateur avant l’action dans la barre d’outils (ou après si toolbar\_pos est positif). Par défaut à False.
- **toolbar\_category** – catégorie de la barre d’outils. Par défaut à None. S’il n’est pas None, cela spécifie la catégorie de la barre d’outils. Si None, par défaut à ActionCategory.VIEW\_TOOLBAR si la catégorie actuelle est ActionCategory.VIEW, sinon à ActionCategory.PANEL\_TOOLBAR.

**Renvoie**

Nouvelle action

**new\_category**(*category* : [ActionCategory](#)) → [Generator](#)[None, None, None]

Gestionnaire de contexte pour créer un nouveau menu.

**Paramètres****category** – Catégorie d'action**Renvoie (Yields)**

None

**new\_menu**(*title* : *str*, *icon\_name* : *str* | *None* = *None*) → [Generator](#)[None, None, None]

Gestionnaire de contexte pour créer un nouveau menu.

**Paramètres**— **title** – Titre du menu— **icon\_name** – Nom de l'icône du menu. Par défaut à None.**Renvoie (Yields)**

None

**selected\_objects\_changed**(*selected\_groups* : *list*[*ObjectGroup*], *selected\_objects* : *list*[*SignalObj* | *ImageObj*]) → None

Mettre à jour les actions en fonction des objets sélectionnés.

**Paramètres**— **selected\_groups** – groupes sélectionnés— **selected\_objects** – objets sélectionnés

## 3.10 Vue des objets

Le module `cdl.core.gui.objectview` fournit des widgets pour afficher des arbres d'objets (signal/image).

---

**Note :** This module provides tree widgets to display signals, images and groups. It is important to note that, by design, the user can only select either individual signals/images or groups, but not both at the same time. This is an important design choice, as it allows to simplify the user experience, and to avoid potential confusion between the two types of selection.

---

### 3.10.1 Simple object tree

**class** `cdl.core.gui.objectview.SimpleObjectTree`(*parent* : *QW.QWidget*, *objmodel* : *ObjectModel*)

Base object handling panel list widget, object (sig/ima) lists

**initialize\_from**(*sobjlist* : [SimpleObjectTree](#)) → None

Init from another SimpleObjectList, without making copies of objects

**iter\_items**(*item* : *QTreeWidgetItem* | *None* = *None*) → [Iterator](#)[*QTreeWidgetItem*]

Recursively iterate over all items

**get\_item\_from\_id**(*item\_id*) → *QTreeWidgetItem*Return *QTreeWidgetItem* from id (stored in item's data)**get\_current\_item\_id**(*object\_only* : *bool* = *False*) → *str* | *None*

Return current item id

**set\_current\_item\_id**(*uuid* : *str*, *extend* : *bool* = *False*) → *None*

Set current item by id

**get\_current\_group\_id**() → *str*

Return current group ID

**get\_sel\_group\_items**() → *list*[*QTreeWidgetItem*]

Return selected group items

**get\_sel\_group\_uuids**() → *list*[*str*]

Return selected group uuids

**get\_sel\_object\_items**() → *list*[*QTreeWidgetItem*]

Return selected object items

**get\_sel\_object\_uuids**(*include\_groups* : *bool* = *False*) → *list*[*str*]

Return selected objects uuids.

**Paramètres**

**include\_groups** – If True, also return objects from selected groups.

**Renvoie**

List of selected objects uuids.

**get\_sel\_objects**(*include\_groups* : *bool* = *False*) → *list*[*SignalObj* | *ImageObj*]

Return selected objects.

If *include\_groups* is True, also return objects from selected groups.

**get\_sel\_groups**() → *list*[*ObjectGroup*]

Return selected groups

**populate\_tree**() → *None*

Populate tree with objects

**update\_tree**() → *None*

Update tree

**add\_group\_item**(*group* : *ObjectGroup*) → *None*

Add group item

**add\_object\_item**(*obj* : *SignalObj* | *ImageObj*, *group\_id* : *str*, *set\_current* : *bool* = *True*) → *None*

Add item

**update\_item**(*uuid* : *str*) → *None*

Update item

**remove\_item**(*oid* : *str*, *refresh* : *bool* = *True*) → *None*

Remove item

**item\_double\_clicked**(*item* : *QTreeWidgetItem*) → *None*

Item was double-clicked : open a pop-up plot dialog

**contextMenuEvent**(*event* : *QContextMenuEvent*) → *None*

Override Qt method

### 3.10.2 Get object dialog

```
class cdl.core.gui.objectview.GetObjectDialog(parent : QW.QWidget, panel : BaseDataPanel, title :
                                         str, comment : str = "", nb_objects : int = 1,
                                         minimum_size : tuple[int, int] | None = None)
```

Dialog box showing groups and objects (signals or images) to select one, or more.

#### Paramètres

- **parent** – parent widget
- **panel** – data panel
- **title** – dialog title
- **comment** – optional dialog comment
- **nb\_objects** – number of objects to select (default : 1)
- **minimum\_size** – minimum size (width, height)

```
get_selected_objects() → list[SignalObj | ImageObj]
```

Return selected objects

### 3.10.3 Vue des objets

```
class cdl.core.gui.objectview.ObjectView(parent : QW.QWidget, objmodel : ObjectModel)
```

Object handling panel list widget, object (sig/ima) lists

```
paintEvent(event)
```

Reimplement Qt method

```
dragEnterEvent(event : QDragEnterEvent) → None
```

Reimplement Qt method

```
dragLeaveEvent(event : QDragLeaveEvent) → None
```

Reimplement Qt method

```
dragMoveEvent(event : QDragMoveEvent) → None
```

Reimplement Qt method

```
get_all_group_uuids() → list[str]
```

Return all group uuids, in a list ordered by group position in the tree

```
get_all_object_uuids() → dict[str, list[str]]
```

Return all object uuids, in a dictionary that maps group uuids to the list of object uuids in each group, in the correct order

```
dropEvent(event : QDropEvent) → None
```

Reimplement Qt method

```
get_current_object() → SignalObj | ImageObj | None
```

Return current object

```
set_current_object(obj : SignalObj | ImageObj) → None
```

Set current object

```
item_selection_changed() → None
```

Refreshing the selection of objects and groups, emitting the SIG\_SELECTION\_CHANGED signal which triggers the update of the object properties panel, the plot items and the actions of the toolbar and menu bar.

This method is called when the user selects or deselects items in the tree. It is also called when the user clicks on an item that was already selected.

This method emits the SIG\_SELECTION\_CHANGED signal.

**select\_objects**(*selection* : *list*[SignalObj | ImageObj | *int* | *str*]) → *None*

Select multiple objects

**Paramètres**

**selection** – list of objects, object numbers (1 to N) or object uuids

**select\_groups**(*groups* : *list*[ObjectGroup | *int* | *str*] | *None* = *None*) → *None*

Select multiple groups

**Paramètres**

**groups** – list of groups, group numbers (1 to N), group names or None (select all groups).

Defaults to None.

**move\_up**()

Move selected objects/groups up

**move\_down**()

Move selected objects/groups down

## 3.11 Gestionnaire de visualisation

Le module `cdl.core.gui.plothandler` fournit des gestionnaires de visualisation pour les panneaux de signaux et d'images, c'est-à-dire des classes gérant les items de visualisation *PlotPy* pour représenter les signaux et les images.

### 3.11.1 Gestionnaire de visualisation de signaux

**class** `cdl.core.gui.plothandler.SignalPlotHandler`(*panel* : *BaseDataPanel*, *plotwidget* : *PlotWidget*)

Objet gérant les items de visualisation de signaux, les dialogues de visualisation et les options de visualisation

**toggle\_anti\_aliasing**(*state* : *bool*) → *None*

Activer/désactiver l'anti-crénelage

**Paramètres**

**state** – si True, activer l'anti-crénelage

**get\_current\_plot\_options**() → *PlotOptions*

Renvoyer les options de visualisation standard pour les signaux/images

**add\_shapes**(*oid* : *str*, *do\_autoscale* : *bool* = *False*) → *None*

Ajouter des items de formes géométriques associés aux résultats calculés et aux annotations, pour l'objet avec l'uuid donné

**cleanup\_dataview**() → *None*

Nettoyer la vue de données

**clear**() → *None*

Effacer les items de visualisation

**get**(*key* : *str*, *default* : *TypePlotItem* | *None* = *None*) → *TypePlotItem* | *None*

Renvoie l'item associé à l'uuid de l'objet. Si la clé n'est pas trouvée, la valeur par défaut est renvoyée si elle est donnée, sinon None est renvoyé.

**get\_obj\_from\_item**(*item* : *TypePlotItem*) → *TypeObj* | *None*

Renvoyer l'objet associé à l'item de visualisation

**Paramètres**

**item** – item de visualisation

**Renvoie**

Objet associé à l'item de visualisation

**reduce\_shown\_oids**(*oids* : *list[str]*) → *list[str]*

Reduce the number of shown objects to visible items only. The base implementation is to show only the first selected item if the option « Show first only » is enabled.

**Paramètres**

**oids** – list of object uuids

**Renvoie**

Reduced list of object uuids

**refresh\_plot**(*what* : *str*, *update\_items* : *bool* = *True*, *force* : *bool* = *False*) → *None*

Actualiser la visualisation

**Paramètres**

— **what** – chaîne décrivant les objets à actualiser. Les valeurs valides sont « selected » (actualiser les objets sélectionnés), « all » (actualiser tous les objets), « existing » (actualiser les items de visualisation existants), ou un uuid d'objet.

— **update\_items** – si *True*, mettre à jour les items. Si *False*, seulement afficher les items (ne pas les mettre à jour, sauf si l'option « Utiliser la plage LUT de l'item de référence » est activée et que plus d'un item est sélectionné). Par défaut à *True*.

— **force** – if *True*, force refresh even if auto refresh is disabled, and refresh all items associated to objects (even the hidden ones, e.g. when selecting multiple images of the same size and position). Defaults to *False*.

**Lève**

**ValueError** – si *what* n'est pas une valeur valide

**remove\_all\_shape\_items**() → *None*

Supprimer toutes les formes géométriques associées aux items de résultats

**remove\_item**(*oid* : *str*) → *None*

Supprimer l'item de visualisation associé à l'uuid de l'objet

**set\_auto\_refresh**(*auto\_refresh* : *bool*) → *None*

Définir le mode d'actualisation automatique.

**Paramètres**

**auto\_refresh** – si *True*, actualiser les items de visualisation automatiquement

**set\_show\_first\_only**(*show\_first\_only* : *bool*) → *None*

Set show first only mode.

**Paramètres**

**show\_first\_only** – if *True*, show only the first selected item

**static update\_item\_according\_to\_ref\_item**(*item* : *TypePlotItem*, *ref\_item* : *TypePlotItem*) → *None*

Mettre à jour l'item de visualisation selon l'item de référence

**update\_resultproperty\_from\_plot\_item**(*item* : *LabelItem*) → *None*

Mettre à jour la propriété de résultat à partir de l'item de visualisation

### 3.11.2 Gestionnaire de visualisation d'images

**class** `cdl.core.gui.plothandler.ImagePlotHandler`(*panel* : `BaseDataPanel`, *plotwidget* : `PlotWidget`)

Objet gérant les items de visualisation d'images, les dialogues de visualisation et les options de visualisation

**static** `update_item_according_to_ref_item`(*item* : `MaskedImageItem`, *ref\_item* : `MaskedImageItem`)  
→ `None`

Mettre à jour l'item de visualisation selon l'item de référence

**reduce\_shown\_oids**(*oids* : `list[str]`) → `list[str]`

Reduce the number of shown objects to visible items only. The base implementation is to show only the first selected item if the option « Show first only » is enabled.

**Paramètres**

**oids** – list of object uuids

**Renvoie**

Reduced list of object uuids

**refresh\_plot**(*what* : `str`, *update\_items* : `bool` = `True`, *force* : `bool` = `False`) → `None`

Actualiser la visualisation

**Paramètres**

— **what** – chaîne décrivant les objets à actualiser. Les valeurs valides sont « selected » (actualiser les objets sélectionnés), « all » (actualiser tous les objets), « existing » (actualiser les items de visualisation existants), ou un uuid d'objet.

— **update\_items** – si `True`, mettre à jour les items. Si `False`, seulement afficher les items (ne pas les mettre à jour, sauf si l'option « Utiliser la plage LUT de l'item de référence » est activée et que plus d'un item est sélectionné). Par défaut à `True`.

— **force** – if `True`, force refresh even if auto refresh is disabled, and refresh all items associated to objects (even the hidden ones, e.g. when selecting multiple images of the same size and position). Defaults to `False`.

**Lève**

**ValueError** – si *what* n'est pas une valeur valide

**cleanup\_dataview**() → `None`

Nettoyer la vue de données

**get\_current\_plot\_options**() → `PlotOptions`

Renvoyer les options de visualisation standard pour les signaux/images

**add\_shapes**(*oid* : `str`, *do\_autoscale* : `bool` = `False`) → `None`

Ajouter des items de formes géométriques associés aux résultats calculés et aux annotations, pour l'objet avec l'uuid donné

**clear**() → `None`

Effacer les items de visualisation

**get**(*key* : `str`, *default* : `TypePlotItem` | `None` = `None`) → `TypePlotItem` | `None`

Renvoie l'item associé à l'uuid de l'objet. Si la clé n'est pas trouvée, la valeur par défaut est renvoyée si elle est donnée, sinon `None` est renvoyé.

**get\_obj\_from\_item**(*item* : `TypePlotItem`) → `TypeObj` | `None`

Renvoyer l'objet associé à l'item de visualisation

**Paramètres**

**item** – item de visualisation

**Renvoie**

Objet associé à l'item de visualisation

**remove\_all\_shape\_items()** → *None*

Supprimer toutes les formes géométriques associées aux items de résultats

**remove\_item**(oid : *str*) → *None*

Supprimer l'item de visualisation associé à l'uuid de l'objet

**set\_auto\_refresh**(auto\_refresh : *bool*) → *None*

Définir le mode d'actualisation automatique.

**Paramètres**

**auto\_refresh** – si True, actualiser les items de visualisation automatiquement

**set\_show\_first\_only**(show\_first\_only : *bool*) → *None*

Set show first only mode.

**Paramètres**

**show\_first\_only** – if True, show only the first selected item

**update\_resultproperty\_from\_plot\_item**(item : *LabelItem*) → *None*

Mettre à jour la propriété de résultat à partir de l'item de visualisation

## 3.12 Editeur de ROI

Le module `cdl.core.gui.roieditor` fournit les widgets de l'éditeur de ROI pour les signaux et les images.

### 3.12.1 Editeur de ROI de signal

**class** `cdl.core.gui.roieditor.SignalROIEditor`(parent : *PlotDialog*, obj : *TypeObj*, extract : *bool*, item : *TypePlotItem* | *None* = *None*)

Editeur de ROI de signal

**get\_obj\_roi\_class**() → *type[SignalROI]*

Get object ROI class

**add\_tools\_to\_plot\_dialog**() → *None*

Add tools to plot dialog

**setup\_widget**() → *None*

Configurer le widget de l'éditeur de ROI

**update\_roi\_titles**()

Mettre à jour les titres des annotations de ROI

### 3.12.2 Editeur de ROI d'image

**class** `cdl.core.gui.roieditor.ImageROIEditor`(parent : *PlotDialog*, obj : *TypeObj*, extract : *bool*, item : *TypePlotItem* | *None* = *None*)

Editeur de ROI d'image

**get\_obj\_roi\_class**() → *type[ImageROI]*

Get object ROI class

**add\_tools\_to\_plot\_dialog()** → *None*

Add tools to plot dialog

**setup\_widget()** → *None*

Configurer le widget de l'éditeur de ROI

**update\_roi\_titles()** → *None*

Mettre à jour les titres des annotations de ROI

## 3.13 Processeur

Le paquet *cdl.core.gui.processor* fournit les **objets processeur** pour les signaux et les images.

Les objets processeur sont le pont entre les modules de calcul (dans *cdl.computation*) et les modules GUI (dans *cdl.core.gui*). Ils sont utilisés pour appeler les fonctions de calcul et pour mettre à jour l'interface utilisateur depuis l'intérieur des objets de panneau de données.

Lors de la mise en œuvre d'une fonction de traitement dans DataLab, les étapes sont généralement les suivantes :

- Ajout d'une action dans le module *cdl.core.gui.actionhandler* pour déclencher la fonction de traitement depuis l'interface utilisateur (par exemple, un élément de menu ou un bouton de barre d'outils).
- Implémentation de la fonction de calcul dans le module *cdl.computation* (qui appellerait éventuellement l'algorithme du module *cdl.algorithms*).
- Implémentation de la méthode de l'objet processeur dans ce paquet pour appeler la fonction de calcul et éventuellement mettre à jour l'interface utilisateur.

Les objets processeur sont organisés en sous-modules en fonction de leur objectif.

Les sous-modules suivants sont disponibles :

- *cdl.core.gui.processor.base* : Fonctions de traitement communes
- *cdl.core.gui.processor.signal* : Fonctions de traitement de signaux
- *cdl.core.gui.processor.image* : Fonctions de traitement d'images

### 3.13.1 Fonctions communes

**class** *cdl.core.gui.processor.base.Worker*

Travailleur de traitement parallèle, pour exécuter des tâches de longue durée dans un processus séparé

**static** **create\_pool()** → *None*

Créer un pool de traitement parallèle

**static** **terminate\_pool**(*wait : bool = False*) → *None*

Terminer le pool de traitement parallèle.

**Paramètres**

**wait** – wait for all tasks to finish. Defaults to False.

**restart\_pool()** → *None*

Terminate and recreate the pool

**run**(*func : Callable, args : tuple[Any]*) → *None*

Run computation.

**Paramètres**

— **func** – function to run

— **args** – arguments

**close()** → *None*  
 Close worker : close pool properly and wait for all tasks to finish

**is\_computation\_finished()** → *bool*  
 Return True if computation is finished.  
**Renvoie**  
 True if computation is finished  
**Type renvoyé**  
*bool*

**get\_result()** → *CompOut*  
 Return computation result.  
**Renvoie**  
 computation result  
**Type renvoyé**  
*CompOut*

**cdl.core.gui.processor.base.is\_pairwise\_mode()** → *bool*  
 Return True if operation mode is pairwise.  
**Renvoie**  
 True if operation mode is pairwise  
**Type renvoyé**  
*bool*

**class cdl.core.gui.processor.base.BaseProcessor**(*panel : SignalPanel | ImagePanel, plotwidget : PlotWidget*)  
 Object handling data processing : operations, processing, analysis.  
**Paramètres**  
 — **panel** – panel  
 — **plotwidget** – plot widget

**close()**  
 Close processor properly

**set\_process\_isolation\_enabled**(*enabled : bool*) → *None*  
 Set process isolation enabled.  
**Paramètres**  
**enabled** – enabled

**has\_param\_defaults**(*paramclass : type[DataSet]*) → *bool*  
 Return True if parameter defaults are available.  
**Paramètres**  
**paramclass** – parameter class  
**Renvoie**  
 True if parameter defaults are available  
**Type renvoyé**  
*bool*

**update\_param\_defaults**(*param : DataSet*) → *None*  
 Update parameter defaults.  
**Paramètres**  
**param** – parameters

**init\_param**(*param* : *DataSet*, *paramclass* : *type[DataSet]*, *title* : *str*, *comment* : *str* | *None* = *None*) → *tuple[bool, DataSet]*

Initialize processing parameters.

**Paramètres**

- **param** – parameter
- **paramclass** – parameter class
- **title** – title
- **comment** – comment

**Renvoie**

Tuple (edit, param) where edit is True if parameters have been edited, False otherwise.

**compute\_11**(*func* : *Callable*, *param* : *DataSet* | *None* = *None*, *paramclass* : *DataSet* | *None* = *None*, *title* : *str* | *None* = *None*, *comment* : *str* | *None* = *None*, *edit* : *bool* | *None* = *None*) → *None*

Compute 11 function : 1 object in → 1 object out.

**Paramètres**

- **func** – function
- **param** – parameter
- **paramclass** – parameter class
- **title** – title
- **comment** – comment
- **edit** – edit parameters

**compute\_1n**(*funcs* : *list[Callable]* | *Callable*, *params* : *list* | *None* = *None*, *title* : *str* | *None* = *None*, *edit* : *bool* | *None* = *None*) → *None*

Compute 1n function : 1 object in → n objects out.

**Paramètres**

- **funcs** – list of functions
- **params** – list of parameters
- **title** – title
- **edit** – edit parameters

**handle\_output**(*compout* : *CompOut*, *context* : *str*, *progress* : *QW.QProgressDialog*) → *SignalObj* | *ImageObj* | *ResultShape* | *ResultProperties* | *None*

Handle computation output : if error, display error message, if warning, display warning message.

**Paramètres**

- **compout** – computation output
- **context** – context (e.g. « Computing : Gaussian filter »)
- **progress** – progress dialog

**Renvoie**

a signal or image object, or a result shape object,  
or None if error

**Type renvoyé**

Output object

**compute\_10**(*func* : *Callable*, *param* : *DataSet* | *None* = *None*, *paramclass* : *DataSet* | *None* = *None*, *title* : *str* | *None* = *None*, *comment* : *str* | *None* = *None*, *edit* : *bool* | *None* = *None*) → *dict[str, ResultShape* | *ResultProperties]*

Compute 10 function : 1 object in → 0 object out (the result of this method is stored in original object's metadata).

**Paramètres**

- **func** – function to execute
- **param** – parameters. Defaults to None.

- **paramclass** – parameters class. Defaults to None.
- **title** – title of progress bar. Defaults to None.
- **comment** – comment. Defaults to None.
- **edit** – if True, edit parameters. Defaults to None.

**Renvoie**

**object uuid, values : ResultShape or**  
ResultProperties objects)

**Type renvoyé**

Dictionary of results (keys

**compute\_n1**(*name* : *str*, *func* : *Callable*, *param* : *DataSet* | *None* = *None*, *paramclass* : *DataSet* | *None* = *None*, *title* : *str* | *None* = *None*, *comment* : *str* | *None* = *None*, *func\_objs* : *Callable* | *None* = *None*, *edit* : *bool* | *None* = *None*) → *None*

Compute n1 function : N(>=2) objects in → 1 object out.

**Paramètres**

- **name** – name of function
- **func** – function to execute
- **param** – parameters. Defaults to None.
- **paramclass** – parameters class. Defaults to None.
- **title** – title of progress bar. Defaults to None.
- **comment** – comment. Defaults to None.
- **func\_objs** – function to execute on objects. Defaults to None.
- **edit** – if True, edit parameters. Defaults to None.

**compute\_n1n**(*obj2* : *Obj* | *list*[*Obj*] | *None*, *obj2\_name* : *str*, *func* : *Callable*, *param* : *gds.DataSet* | *None* = *None*, *paramclass* : *gds.DataSet* | *None* = *None*, *title* : *str* | *None* = *None*, *comment* : *str* | *None* = *None*, *edit* : *bool* | *None* = *None*) → *None*

Compute n1n function : N(>=1) objects + 1 object in → N objects out.

Examples : subtract, divide

**Paramètres**

- **obj2** – second object (or list of objects in case of pairwise operation mode)
- **obj2\_name** – name of second object
- **func** – function to execute
- **param** – parameters. Defaults to None.
- **paramclass** – parameters class. Defaults to None.
- **title** – title of progress bar. Defaults to None.
- **comment** – comment. Defaults to None.
- **edit** – if True, edit parameters. Defaults to None.

**abstract compute\_arithmetic**(*obj2* : *Obj* | *None* = *None*, *param* : *ArithmeticParam* | *None* = *None*) → *None*

Compute arithmetic operation

**abstract compute\_sum**() → *None*

Compute sum

**abstract compute\_normalize**(*param* : *NormalizeParam* | *None* = *None*) → *None*

Normalize data

**abstract compute\_average**() → *None*

Compute average

**abstract compute\_product**() → *None*

Compute product

**abstract compute\_difference**(obj2 : Obj | list[Obj] | None = None) → None  
Compute difference

**abstract compute\_quadratic\_difference**(obj2 : Obj | list[Obj] | None = None) → None  
Compute quadratic difference

**abstract compute\_division**(obj2 : Obj | list[Obj] | None = None) → None  
Compute division

**abstract compute\_swap\_axes**() → None  
Swap data axes

**abstract compute\_abs**() → None  
Compute absolute value

**abstract compute\_re**() → None  
Compute real part

**abstract compute\_im**() → None  
Compute imaginary part

**abstract compute\_astype**() → None  
Convert data type

**abstract compute\_log10**() → None  
Compute Log10

**abstract compute\_exp**() → None  
Compute exponential

**abstract compute\_calibration**(param=None) → None  
Compute data linear calibration

**abstract compute\_clip**(param : ClipParam | None = None) → None  
Compute maximum data clipping

**abstract compute\_gaussian\_filter**(param : GaussianParam | None = None) → None  
Compute gaussian filter

**abstract compute\_moving\_average**(param : MovingAverageParam | None = None) → None  
Compute moving average

**abstract compute\_moving\_median**(param : MovingMedianParam | None = None) → None  
Compute moving median

**abstract compute\_wiener**() → None  
Compute Wiener filter

**abstract compute\_fft**() → None  
Compute iFFT

**abstract compute\_ifft**() → None  
Compute FFT

**abstract compute\_addition\_constant**(param : ConstantParam) → None  
Compute sum with a constant

**abstract compute\_difference\_constant**(param : ConstantParam) → None  
Compute difference with a constant

**abstract compute\_product\_constant**(*param* : *ConstantParam*) → *None*  
 Compute product with a constant

**abstract compute\_division\_constant**(*param* : *ConstantParam*) → *None*  
 Compute division by a constant

**compute\_roi\_extraction**(*roi* : *TypeROI* | *None* = *None*) → *None*  
 Extract Region Of Interest (ROI) from data with :  
 — *cdl.computation.image.extract\_single\_roi()* for single ROI  
 — *cdl.computation.image.extract\_multiple\_roi()* for multiple ROIs

**edit\_regions\_of\_interest**(*extract* : *bool* = *False*) → *TypeROI* | *None*  
 Define Region Of Interest (ROI).

**Paramètres**  
**extract** – If True, ROI is extracted from data. Defaults to False.

**Renvoie**  
 ROI object or None if ROI dialog has been canceled.

**delete\_regions\_of\_interest**() → *None*  
 Delete Regions Of Interest

**abstract compute\_stats**() → *dict*[*str*, *ResultShape*]  
 Compute data statistics

### 3.13.2 Signal processing features

**class** *cdl.core.gui.processor.signal.SignalProcessor*(*panel* : *SignalPanel* | *ImagePanel*, *plotwidget* : *PlotWidget*)

Object handling signal processing : operations, processing, analysis

**compute\_sum**() → *None*  
 Compute sum with *cdl.computation.signal.compute\_addition()*

**compute\_addition\_constant**(*param* : *ConstantParam* | *None* = *None*) → *None*  
 Compute sum with a constant with *cdl.computation.signal.compute\_addition\_constant()*

**compute\_average**() → *None*  
 Compute average with *cdl.computation.signal.compute\_addition()* and divide by the number of signals

**compute\_product**() → *None*  
 Compute product with *cdl.computation.signal.compute\_product()*

**compute\_product\_constant**(*param* : *ConstantParam* | *None* = *None*) → *None*  
 Compute product with a constant with *cdl.computation.signal.compute\_product\_constant()*

**compute\_swap\_axes**() → *None*  
 Swap data axes with *cdl.computation.signal.compute\_swap\_axes()*

**compute\_abs**() → *None*  
 Compute absolute value with *cdl.computation.signal.compute\_abs()*

**compute\_re**() → *None*  
 Compute real part with *cdl.computation.signal.compute\_re()*

**compute\_im()** → None  
Compute imaginary part with `cdl.computation.signal.compute_im()`

**compute\_astype**(*param* : DataTypeSParam | None = None) → None  
Convert data type with `cdl.computation.signal.compute_astype()`

**compute\_log10()** → None  
Compute Log10 with `cdl.computation.signal.compute_log10()`

**compute\_exp()** → None  
Compute Log10 with `cdl.computation.signal.compute_exp()`

**compute\_sqrt()** → None  
Compute square root with `cdl.computation.signal.compute_sqrt()`

**compute\_power**(*param* : PowerParam | None = None) → None  
Compute power with `cdl.computation.signal.compute_power()`

**compute\_arithmetic**(*obj2* : SignalObj | None = None, *param* : ArithmeticParam | None = None) → None  
Compute arithmetic operation between two signals with `cdl.computation.signal.compute_arithmetic()`

**compute\_difference**(*obj2* : SignalObj | list[SignalObj] | None = None) → None  
Compute difference between two signals with `cdl.computation.signal.compute_difference()`

**compute\_difference\_constant**(*param* : ConstantParam | None = None) → None  
Compute difference with a constant with `cdl.computation.signal.compute_difference_constant()`

**compute\_quadratic\_difference**(*obj2* : SignalObj | list[SignalObj] | None = None) → None  
Compute quadratic difference between two signals with `cdl.computation.signal.compute_quadratic_difference()`

**compute\_division**(*obj2* : SignalObj | list[SignalObj] | None = None) → None  
Compute division between two signals with `cdl.computation.signal.compute_division()`

**compute\_division\_constant**(*param* : ConstantParam | None = None) → None  
Compute division by a constant with `cdl.computation.signal.compute_division_constant()`

**compute\_peak\_detection**(*param* : PeakDetectionParam | None = None) → None  
Detect peaks from data with `cdl.computation.signal.compute_peak_detection()`

**compute\_reverse\_x()** → None  
Reverse X axis with `cdl.computation.signal.compute_reverse_x()`

**compute\_normalize**(*param* : NormalizeParam | None = None) → None  
Normalize data with `cdl.computation.signal.compute_normalize()`

**compute\_derivative**() → None  
Compute derivative with `cdl.computation.signal.compute_derivative()`

**compute\_integral**() → None  
Compute integral with `cdl.computation.signal.compute_integral()`

**compute\_calibration**(*param* : XYCalibrateParam | None = None) → None  
Compute data linear calibration with `cdl.computation.signal.compute_calibration()`

**compute\_clip**(*param* : ClipParam | None = None) → None  
Compute maximum data clipping with `cdl.computation.signal.compute_clip()`

```

compute_offset_correction(param : ROIIDParam | None = None) → None
    Compute offset correction with cdl.computation.signal.compute_offset_correction()

compute_gaussian_filter(param : GaussianParam | None = None) → None
    Compute gaussian filter with cdl.computation.signal.compute_gaussian_filter()

compute_moving_average(param : MovingAverageParam | None = None) → None
    Compute moving average with cdl.computation.signal.compute_moving_average()

compute_moving_median(param : MovingMedianParam | None = None) → None
    Compute moving median with cdl.computation.signal.compute_moving_median()

compute_wiener() → None
    Compute Wiener filter with cdl.computation.signal.compute_wiener()

compute_lowpass(param : LowPassFilterParam | None = None) → None
    Compute high-pass filter with cdl.computation.signal.compute_filter()

compute_highpass(param : HighPassFilterParam | None = None) → None
    Compute high-pass filter with cdl.computation.signal.compute_filter()

compute_bandpass(param : BandPassFilterParam | None = None) → None
    Compute band-pass filter with cdl.computation.signal.compute_filter()

compute_bandstop(param : BandStopFilterParam | None = None) → None
    Compute band-stop filter with cdl.computation.signal.compute_filter()

compute_fft(param : FFTParam | None = None) → None
    Compute FFT with cdl.computation.signal.compute_fft()

compute_ifft(param : FFTParam | None = None) → None
    Compute iFFT with cdl.computation.signal.compute_ifft()

compute_magnitude_spectrum(param : SpectrumParam | None = None) → None
    Compute magnitude spectrum with cdl.computation.signal.compute_magnitude_spectrum()

compute_phase_spectrum() → None
    Compute phase spectrum with cdl.computation.signal.compute_phase_spectrum()

compute_psd(param : SpectrumParam | None = None) → None
    Compute power spectral density with cdl.computation.signal.compute_psd()

compute_interpolation(obj2 : SignalObj | None = None, param : InterpolationParam | None = None)
    Compute interpolation with cdl.computation.signal.compute_interpolation()

compute_resampling(param : ResamplingParam | None = None)
    Compute resampling with cdl.computation.signal.compute_resampling()

compute_detrending(param : DetrendingParam | None = None)
    Compute detrending with cdl.computation.signal.compute_detrending()

compute_convolution(obj2 : SignalObj | None = None) → None
    Compute convolution with cdl.computation.signal.compute_convolution()

compute_windowing(param : WindowingParam | None = None) → None
    Compute windowing with cdl.computation.signal.compute_windowing()

compute_polyfit(param : PolynomialFitParam | None = None) → None
    Compute polynomial fitting curve

```

**compute\_fit**(*title* : *str*, *fitdlgfunc* : *Callable*) → *None*  
 Compute fitting curve using an interactive dialog  
**Paramètres**  
 — **title** – Title of the dialog  
 — **fitdlgfunc** – Fitting dialog function

**compute\_multigaussianfit**() → *None*  
 Compute multi-Gaussian fitting curve using an interactive dialog

**compute\_fwhm**(*param* : *FWHMParam* | *None* = *None*) → *dict*[*str*, *ResultShape*]  
 Compute FWHM with *cdl.computation.signal.compute\_fwhm()*

**compute\_fw1e2**() → *dict*[*str*, *ResultShape*]  
 Compute FW at  $1/e^2$  with *cdl.computation.signal.compute\_fw1e2()*

**compute\_stats**() → *dict*[*str*, *ResultProperties*]  
 Compute data statistics with *cdl.computation.signal.compute\_stats()*

**compute\_histogram**(*param* : *HistogramParam* | *None* = *None*) → *dict*[*str*, *ResultShape*]  
 Compute histogram with *cdl.computation.signal.compute\_histogram()*

**compute\_contrast**() → *dict*[*str*, *ResultProperties*]  
 Compute contrast with *cdl.computation.signal.compute\_contrast()*

**compute\_x\_at\_minmax**() → *dict*[*str*, *ResultProperties*]  
 Compute x at min/max with *cdl.computation.signal.compute\_x\_at\_minmax()*

**compute\_sampling\_rate\_period**() → *dict*[*str*, *ResultProperties*]  
 Compute sampling rate and period (mean and std) with *cdl.computation.signal.compute\_sampling\_rate\_period()*

**compute\_bandwidth\_3db**() → *None*  
 Compute bandwidth at -3dB with *cdl.computation.signal.compute\_bandwidth\_3db()*

**compute\_dynamic\_parameters**(*param* : *DynamicParam* | *None* = *None*) → *dict*[*str*, *ResultProperties*]  
 Compute Dynamic Parameters (ENOB, SINAD, THD, SFDR, SNR) with *cdl.computation.signal.compute\_dynamic\_parameters()*

### 3.13.3 Image processing features

**class** *cdl.core.gui.processor.image.ImageProcessor*(*panel* : *SignalPanel* | *ImagePanel*, *plotwidget* : *PlotWidget*)  
 Object handling image processing : operations, processing, analysis

**compute\_normalize**(*param* : *NormalizeParam* | *None* = *None*) → *None*  
 Normalize data with *cdl.computation.image.compute\_normalize()*

**compute\_sum**() → *None*  
 Compute sum with *cdl.computation.image.compute\_addition()*

**compute\_addition\_constant**(*param* : *ConstantParam* | *None* = *None*) → *None*  
 Compute sum with a constant using *cdl.computation.image.compute\_addition\_constant()*

**compute\_average**() → *None*  
 Compute average with *cdl.computation.image.compute\_addition()* and dividing by the number of images

**compute\_product()** → None  
 Compute product with `cdl.computation.image.compute_product()`

**compute\_product\_constant**(*param* : ConstantParam | None = None) → None  
 Compute product with a constant using `cdl.computation.image.compute_product_constant()`

**compute\_logp1**(*param* : LogP1Param | None = None) → None  
 Compute base 10 logarithm using `cdl.computation.image.compute_logp1()`

**compute\_rotate**(*param* : RotateParam | None = None) → None  
 Rotate data arbitrarily using `cdl.computation.image.compute_rotate()`

**compute\_rotate90**() → None  
 Rotate data 90° with `cdl.computation.image.compute_rotate90()`

**compute\_rotate270**() → None  
 Rotate data 270° with `cdl.computation.image.compute_rotate270()`

**compute\_fliph**() → None  
 Flip data horizontally using `cdl.computation.image.compute_fliph()`

**compute\_flipv**() → None  
 Flip data vertically with `cdl.computation.image.compute_flipv()`

**distribute\_on\_grid**(*param* : GridParam | None = None) → None  
 Distribute images on a grid

**reset\_positions**() → None  
 Reset image positions

**compute\_resize**(*param* : ResizeParam | None = None) → None  
 Resize image with `cdl.computation.image.compute_resize()`

**compute\_binning**(*param* : BinningParam | None = None) → None  
 Binning image with `cdl.computation.image.compute_binning()`

**compute\_line\_profile**(*param* : LineProfileParam | None = None) → None  
 Compute profile along a vertical or horizontal line with `cdl.computation.image.compute_line_profile()`

**compute\_segment\_profile**(*param* : SegmentProfileParam | None = None)  
 Compute profile along a segment with `cdl.computation.image.compute_segment_profile()`

**compute\_average\_profile**(*param* : AverageProfileParam | None = None) → None  
 Compute average profile with `cdl.computation.image.compute_average_profile()`

**compute\_radial\_profile**(*param* : RadialProfileParam | None = None) → None  
 Compute radial profile with `cdl.computation.image.compute_radial_profile()`

**compute\_histogram**(*param* : HistogramParam | None = None) → None  
 Compute histogram with `cdl.computation.image.compute_histogram()`

**compute\_swap\_axes**() → None  
 Swap data axes with `cdl.computation.image.compute_swap_axes()`

**compute\_abs**() → None  
 Compute absolute value with `cdl.computation.image.compute_abs()`

**compute\_re()** → None  
Compute real part with `cdl.computation.image.compute_re()`

**compute\_im()** → None  
Compute imaginary part with `cdl.computation.image.compute_im()`

**compute\_astype**(*param* : `DataTypeIParam` | *None* = *None*) → None  
Convert data type with `cdl.computation.image.compute_astype()`

**compute\_log10()** → None  
Compute Log10 with `cdl.computation.image.compute_log10()`

**compute\_exp()** → None  
Compute Log10 with `cdl.computation.image.compute_exp()`

**compute\_arithmetic**(*obj2* : `ImageObj` | *None* = *None*, *param* : `ArithmeticParam` | *None* = *None*) → None  
Compute arithmetic operation between two images with `cdl.computation.image.compute_arithmetic()`

**compute\_difference**(*obj2* : `ImageObj` | *list*[`ImageObj`] | *None* = *None*) → None  
Compute difference between two images with `cdl.computation.image.compute_difference()`

**compute\_difference\_constant**(*param* : `ConstantParam` | *None* = *None*) → None  
Compute difference with a constant with `cdl.computation.image.compute_difference_constant()`

**compute\_quadratic\_difference**(*obj2* : `ImageObj` | *list*[`ImageObj`] | *None* = *None*) → None  
Compute quadratic difference between two images with `cdl.computation.image.compute_quadratic_difference()`

**compute\_division**(*obj2* : `ImageObj` | *list*[`ImageObj`] | *None* = *None*) → None  
Compute division between two images with `cdl.computation.image.compute_division()`

**compute\_division\_constant**(*param* : `ConstantParam` | *None* = *None*) → None  
Compute division by a constant with `cdl.computation.image.compute_division_constant()`

**compute\_flatfield**(*obj2* : `ImageObj` | *None* = *None*, *param* : `FlatFieldParam` | *None* = *None*) → None  
Compute flat field correction with `cdl.computation.image.compute_flatfield()`

**compute\_calibration**(*param* : `ZCalibrateParam` | *None* = *None*) → None  
Compute data linear calibration with `cdl.computation.image.compute_calibration()`

**compute\_clip**(*param* : `ClipParam` | *None* = *None*) → None  
Compute maximum data clipping with `cdl.computation.image.compute_clip()`

**compute\_offset\_correction**(*param* : `ROI2DParam` | *None* = *None*) → None  
Compute offset correction with `cdl.computation.image.compute_offset_correction()`

**compute\_gaussian\_filter**(*param* : `GaussianParam` | *None* = *None*) → None  
Compute gaussian filter with `cdl.computation.image.compute_gaussian_filter()`

**compute\_moving\_average**(*param* : `MovingAverageParam` | *None* = *None*) → None  
Compute moving average with `cdl.computation.image.compute_moving_average()`

**compute\_moving\_median**(*param* : `MovingMedianParam` | *None* = *None*) → None  
Compute moving median with `cdl.computation.image.compute_moving_median()`

**compute\_wiener()** → None  
Compute Wiener filter with `cdl.computation.image.compute_wiener()`

```

compute_fft(param : FFTParam | None = None) → None
    Compute FFT with cdl.computation.image.compute_fft()

compute_ifft(param : FFTParam | None = None) → None
    Compute iFFT with cdl.computation.image.compute_ifft()

compute_magnitude_spectrum(param : SpectrumParam | None = None) → None
    Compute magnitude spectrum with cdl.computation.image.compute_magnitude_spectrum()

compute_phase_spectrum() → None
    Compute phase spectrum with cdl.computation.image.compute_phase_spectrum()

compute_psd(param : SpectrumParam | None = None) → None
    Compute Power Spectral Density (PSD) with cdl.computation.image.compute_psd()

compute_butterworth(param : ButterworthParam | None = None) → None
    Compute Butterworth filter with cdl.computation.image.compute_butterworth()

compute_threshold(param : ThresholdParam | None = None) → None
    Compute parametric threshold with cdl.computation.image.threshold.compute_threshold()

compute_threshold_isodata() → None
    Compute threshold using Isodata algorithm with cdl.computation.image.threshold.compute_threshold_isodata()

compute_threshold_li() → None
    Compute threshold using Li algorithm with cdl.computation.image.threshold.compute_threshold_li()

compute_threshold_mean() → None
    Compute threshold using Mean algorithm with cdl.computation.image.threshold.compute_threshold_mean()

compute_threshold_minimum() → None
    Compute threshold using Minimum algorithm with cdl.computation.image.threshold.compute_threshold_minimum()

compute_threshold_otsu() → None
    Compute threshold using Otsu algorithm with cdl.computation.image.threshold.compute_threshold_otsu()

compute_threshold_triangle() → None
    Compute threshold using Triangle algorithm with cdl.computation.image.threshold.compute_threshold_triangle()

compute_threshold_yen() → None
    Compute threshold using Yen algorithm with cdl.computation.image.threshold.compute_threshold_yen()

compute_all_threshold() → None
    Compute all threshold algorithms using the following functions :
    — cdl.computation.image.threshold.compute_threshold_isodata()
    — cdl.computation.image.threshold.compute_threshold_li()
    — cdl.computation.image.threshold.compute_threshold_mean()
    — cdl.computation.image.threshold.compute_threshold_minimum()
    — cdl.computation.image.threshold.compute_threshold_otsu()
    — cdl.computation.image.threshold.compute_threshold_triangle()
    — cdl.computation.image.threshold.compute_threshold_yen()

```

**compute\_adjust\_gamma**(*param* : AdjustGammaParam | None = None) → None  
 Compute gamma correction with `cdl.computation.image.exposure.compute_adjust_gamma()`

**compute\_adjust\_log**(*param* : AdjustLogParam | None = None) → None  
 Compute log correction with `cdl.computation.image.exposure.compute_adjust_log()`

**compute\_adjust\_sigmoid**(*param* : AdjustSigmoidParam | None = None) → None  
 Compute sigmoid correction with `cdl.computation.image.exposure.compute_adjust_sigmoid()`

**compute\_rescale\_intensity**(*param* : RescaleIntensityParam | None = None) → None  
 Rescale image intensity levels with `py:func`cdl.computation.image.exposure.compute_rescale_intensity``

**compute\_equalize\_hist**(*param* : EqualizeHistParam | None = None) → None  
 Histogram equalization with `cdl.computation.image.exposure.compute_equalize_hist()`

**compute\_equalize\_adapthist**(*param* : EqualizeAdaptHistParam | None = None) → None  
 Adaptive histogram equalization with `cdl.computation.image.exposure.compute_equalize_adapthist()`

**compute\_denoise\_tv**(*param* : DenoiseTVParam | None = None) → None  
 Compute Total Variation denoising with `cdl.computation.image.restoration.compute_denoise_tv()`

**compute\_denoise\_bilateral**(*param* : DenoiseBilateralParam | None = None) → None  
 Compute bilateral filter denoising with `cdl.computation.image.restoration.compute_denoise_bilateral()`

**compute\_denoise\_wavelet**(*param* : DenoiseWaveletParam | None = None) → None  
 Compute Wavelet denoising with `cdl.computation.image.restoration.compute_denoise_wavelet()`

**compute\_denoise\_tophat**(*param* : MorphologyParam | None = None) → None  
 Denoise using White Top-Hat with `cdl.computation.image.restoration.compute_denoise_tophat()`

**compute\_all\_denoise**(*params* : list | None = None) → None  
 Compute all denoising filters using the following functions :  
 — `cdl.computation.image.restoration.compute_denoise_tv()`  
 — `cdl.computation.image.restoration.compute_denoise_bilateral()`  
 — `cdl.computation.image.restoration.compute_denoise_wavelet()`  
 — `cdl.computation.image.restoration.compute_denoise_tophat()`

**compute\_white\_tophat**(*param* : MorphologyParam | None = None) → None  
 Compute White Top-Hat with `cdl.computation.image.morphology.compute_white_tophat()`

**compute\_black\_tophat**(*param* : MorphologyParam | None = None) → None  
 Compute Black Top-Hat with `cdl.computation.image.morphology.compute_black_tophat()`

**compute\_erosion**(*param* : MorphologyParam | None = None) → None  
 Compute Erosion with `cdl.computation.image.morphology.compute_erosion()`

**compute\_dilation**(*param* : MorphologyParam | None = None) → None  
 Compute Dilation with `cdl.computation.image.morphology.compute_dilation()`

**compute\_opening**(*param* : MorphologyParam | None = None) → None  
 Compute morphological opening with `cdl.computation.image.morphology.compute_opening()`

**compute\_closing**(*param* : MorphologyParam | None = None) → None  
 Compute morphological closing with `cdl.computation.image.morphology.compute_closing()`

**compute\_all\_morphology**(*param* : MorphologyParam | None = None) → None  
 Compute all morphology filters using the following functions :  
 — `cdl.computation.image.morphology.compute_white_tophat()`  
 — `cdl.computation.image.morphology.compute_black_tophat()`  
 — `cdl.computation.image.morphology.compute_erosion()`  
 — `cdl.computation.image.morphology.compute_dilation()`  
 — `cdl.computation.image.morphology.compute_opening()`  
 — `cdl.computation.image.morphology.compute_closing()`

**compute\_canny**(*param* : CannyParam | None = None) → None  
 Compute Canny filter with `cdl.computation.image.edges.compute_canny()`

**compute\_roberts**() → None  
 Compute Roberts filter with `cdl.computation.image.edges.compute_roberts()`

**compute\_prewitt**() → None  
 Compute Prewitt filter with `cdl.computation.image.edges.compute_prewitt()`

**compute\_prewitt\_h**() → None  
 Compute Prewitt filter (horizontal) with `cdl.computation.image.edges.compute_prewitt_h()`

**compute\_prewitt\_v**() → None  
 Compute Prewitt filter (vertical) with `cdl.computation.image.edges.compute_prewitt_v()`

**compute\_sobel**() → None  
 Compute Sobel filter with `cdl.computation.image.edges.compute_sobel()`

**compute\_sobel\_h**() → None  
 Compute Sobel filter (horizontal) with `cdl.computation.image.edges.compute_sobel_h()`

**compute\_sobel\_v**() → None  
 Compute Sobel filter (vertical) with `cdl.computation.image.edges.compute_sobel_v()`

**compute\_scharr**() → None  
 Compute Scharr filter with `cdl.computation.image.edges.compute_scharr()`

**compute\_scharr\_h**() → None  
 Compute Scharr filter (horizontal) with `cdl.computation.image.edges.compute_scharr_h()`

**compute\_scharr\_v**() → None  
 Compute Scharr filter (vertical) with `cdl.computation.image.edges.compute_scharr_v()`

**compute\_farid**() → None  
 Compute Farid filter with `cdl.computation.image.edges.compute_farid()`

**compute\_farid\_h**() → None  
 Compute Farid filter (horizontal) with `cdl.computation.image.edges.compute_farid_h()`

**compute\_farid\_v**() → None  
 Compute Farid filter (vertical) with `cdl.computation.image.edges.compute_farid_v()`

**compute\_laplace**() → None  
 Compute Laplace filter with `cdl.computation.image.edges.compute_laplace()`

**compute\_all\_edges**() → None  
 Compute all edges filters using the following functions :

```
— cdl.computation.image.edges.compute_roberts()
— cdl.computation.image.edges.compute_prewitt()
— cdl.computation.image.edges.compute_prewitt_h()
— cdl.computation.image.edges.compute_prewitt_v()
— cdl.computation.image.edges.compute_sobel()
— cdl.computation.image.edges.compute_sobel_h()
— cdl.computation.image.edges.compute_sobel_v()
— cdl.computation.image.edges.compute_scharr()
— cdl.computation.image.edges.compute_scharr_h()
— cdl.computation.image.edges.compute_scharr_v()
— cdl.computation.image.edges.compute_farid()
— cdl.computation.image.edges.compute_farid_h()
— cdl.computation.image.edges.compute_farid_v()
— cdl.computation.image.edges.compute_laplace()
```

**compute\_stats()** → dict[str, *ResultProperties*]

Compute data statistics with `cdl.computation.image.compute_stats()`

**compute\_centroid()** → dict[str, *ResultShape*]

Compute image centroid with `cdl.computation.image.compute_centroid()`

**compute\_enclosing\_circle()** → dict[str, *ResultShape*]

Compute minimum enclosing circle with `cdl.computation.image.compute_enclosing_circle()`

**compute\_peak\_detection()**(*param* : *Peak2DDetectionParam* | *None* = *None*) → dict[str, *ResultShape*]

Compute 2D peak detection with `cdl.computation.image.compute_peak_detection()`

**compute\_contour\_shape()**(*param* : *ContourShapeParam* | *None* = *None*) → dict[str, *ResultShape*]

Compute contour shape fit with `cdl.computation.image.detection.compute_contour_shape()`

**compute\_hough\_circle\_peaks()**(*param* : *HoughCircleParam* | *None* = *None*) → dict[str, *ResultShape*]

Compute peak detection based on a circle Hough transform with `cdl.computation.image.compute_hough_circle_peaks()`

**compute\_blob\_dog()**(*param* : *BlobDOGParam* | *None* = *None*) → dict[str, *ResultShape*]

Compute blob detection using Difference of Gaussian method with `cdl.computation.image.detection.compute_blob_dog()`

**compute\_blob\_doh()**(*param* : *BlobDOHParam* | *None* = *None*) → dict[str, *ResultShape*]

Compute blob detection using Determinant of Hessian method with `cdl.computation.image.detection.compute_blob_doh()`

**compute\_blob\_log()**(*param* : *BlobLOGParam* | *None* = *None*) → dict[str, *ResultShape*]

Compute blob detection using Laplacian of Gaussian method with `cdl.computation.image.detection.compute_blob_log()`

**compute\_blob\_opencv()**(*param* : *BlobOpenCVParam* | *None* = *None*) → dict[str, *ResultShape*]

Compute blob detection using OpenCV with `cdl.computation.image.detection.compute_blob_opencv()`

## 3.14 Docks

Le module `cdl.core.gui.docks` fournit les widgets dockables pour la fenêtre principale de DataLab.

### 3.14.1 Widget de visualisation

**class** `cdl.core.gui.docks.DataLabPlotWidget`(*plot\_type* : *PlotType*)

DataLab PlotWidget

Cette classe est une sous-classe de `plotpy.plot.PlotWidget` qui fournit un widget personnalisé pour DataLab, avec un ensemble spécifique d'outils et une apparence personnalisée.

**Paramètres**

**plot\_type** – Type de visualisation

**register\_tools()** → *None*

Inscrire les outils de visualisation en fonction du type de visualisation

### 3.14.2 Widget de visualisation dockable

**class** `cdl.core.gui.docks.DockablePlotWidget`(*parent* : *QWidget*, *plot\_type* : *PlotType*)

Widget de visualisation docké

**Paramètres**

— **parent** – Widget parent

— **plot\_type** – Type de visualisation

**setup\_layout()** → *None*

Configurer la disposition

**update\_toolbar\_position()** → *None*

Mettre à jour la position de la barre d'outils

**setup\_plotwidget()** → *None*

Configurer le widget de visualisation

**update\_color\_mode()** → *None*

Update plot widget styles according to application color mode

**get\_plot()** → *BasePlot*

Renvoyer l'instance de visualisation

**update\_watermark**(*plot* : *BasePlot*) → *None*

Mettre à jour la visibilité du filigrane

**visibility\_changed**(*enable* : *bool*) → *None*

La visibilité du DockWidget a changé

## 3.15 E/S HDF5

Le module `cdl.core.gui.h5io` fournit l'ouverture/enregistrement de fichiers HDF5 dans/depuis le modèle de données de DataLab/fenêtre principale.

**class** `cdl.core.gui.h5io.H5InputOutput`(*mainwindow* : `CDLMainWindow`)

Objet gérant l'ouverture/enregistrement de fichiers HDF5 dans/depuis le modèle de données de DataLab/fenêtre principale

**Paramètres**

**mainwindow** – Fenêtre principale

**save\_file**(*filename* : *str*) → `None`

Enregistrer tous les signaux et images du modèle DataLab dans un fichier HDF5

**open\_file**(*filename* : *str*, *import\_all* : *bool*, *reset\_all* : *bool*) → `None`

Ouvrir un fichier HDF5

**import\_files**(*filenames* : *list[str]*, *import\_all* : *bool*, *reset\_all* : *bool*) → `None`

Importer des fichiers HDF5

**import\_dataset\_from\_file**(*filename* : *str*, *dsetname* : *str*) → `None`

Importer un jeu de données depuis un fichier HDF5

DataLab est **votre** plateforme. Si vous souhaitez qu'elle s'améliore, vous **pouvez** contribuer au projet, que vous soyez développeur ou non.

Il existe de nombreuses façons de contribuer au projet DataLab, en fonction du temps dont vous disposez, de votre expérience avec les projets open source et de vos compétences.

### 4.1 Partagez vos idées et vos expériences

Outre les rapports d'anomalies classiques et les demandes de fonctionnalités, vous pouvez partager vos idées et vos expériences pour améliorer DataLab. En particulier, nous sommes très intéressés par vos commentaires sur la documentation et les tutoriels. De plus, si vous avez un cas d'utilisation que vous souhaitez partager avec la communauté, faites-le nous savoir.

Sans coder une seule ligne, vous pouvez contribuer au projet DataLab en :

- [Signaler une anomalie](#)
- [Suggérer une amélioration](#)
- [Suggérer un sujet de documentation](#)
- [Suggérer un sujet de tutoriel](#)

### 4.2 Partagez vos connaissances scientifiques/techniques

Vos connaissances techniques ou scientifiques sont également très précieuses pour nous, car vous pouvez contribuer directement à la documentation ou aux tutoriels. Ou, mieux encore, si vous souhaitez rédiger un tutoriel, nous serons heureux de vous aider.

Encore une fois sans coder, vous pouvez contribuer au projet DataLab en :

- [Ecrire de la documentation](#)
- [Ecrire un tutoriel](#)

## 4.3 Contribuer aux nouvelles fonctionnalités

Même si vous n'êtes pas développeur, vous pouvez contribuer au projet en :

- Testant de nouvelles fonctionnalités
- Ecrivant et soumettant de nouveaux plugins
- Ecrivant et soumettant des macro-commandes

## 4.4 Développer de nouvelles fonctionnalités

Si vous êtes développeur, vous pouvez contribuer au cœur du projet en corrigeant des anomalies ou en implémentant de nouvelles fonctionnalités.

Voir la section *Contribuer au code* pour plus d'informations.

### 4.4.1 Contribuer au code

Cette page explique comment vous pouvez contribuer au code du projet.

**Voir aussi :**

Les règles de codage sont présentées dans la page *Règles de codage*.

#### Forker le projet

La première étape est de forker le projet sur GitHub. Vous pouvez le faire en visitant le projet DataLab et en cliquant sur le bouton « Fork » en haut à droite de la [page GitHub du projet](#).

Une fois que vous avez forké le projet, vous aurez une copie du projet dans votre propre compte GitHub. Ensuite, vous pouvez cloner le projet sur votre ordinateur et commencer à travailler dessus.

#### Soumettre une pull request

Dès que vous avez apporté des modifications, vous pouvez soumettre une pull request au projet original. Pour ce faire, allez sur votre projet forké sur GitHub et cliquez sur le bouton « Pull request » en haut à droite de la page.

Ensuite vous devrez remplir un formulaire pour décrire votre pull request. Une fois que vous avez soumis la pull request, les mainteneurs du projet examineront vos modifications et les fusionneront s'ils sont satisfaits.

Lors du processus de revue, les mainteneurs du projet vérifieront que votre code suit les règles de codage et qu'il ne casse pas les tests existants. Si votre code ne suit pas les directives de codage, vous devrez le corriger avant que votre pull request puisse être fusionnée.

## 4.4.2 Règles de codage

### Règles de codage génériques

Nous suivons le style de codage [PEP 8](#).

En particulier, nous sommes particulièrement stricts sur les directives suivantes :

- Limiter toutes les lignes à un maximum de 79 caractères.
- Respecter les conventions de nommage (classes, fonctions, variables, etc.).
- Utiliser des exceptions spécifiques au lieu de [Exception](#).

Pour faire respecter ces directives, les outils suivants sont obligatoires :

- [ruff](#) pour le formatage du code et l'analyse statique du code.
- [pylint](#) pour l'analyse statique du code.

#### ruff

Si vous utilisez [Visual Studio Code](#), les paramètres du projet formateront automatiquement votre code avec *ruff* à l'enregistrement (vous pouvez également exécuter la tâche « Run Ruff » pour exécuter *ruff* sur le projet).

Pour exécuter *ruff*, exécutez la commande suivante :

```
ruff check
```

#### pylint

Pour exécuter *pylint*, exécutez la commande suivante :

```
pylint datalab
```

Si vous utilisez [Visual Studio Code](#) sur Windows, vous pouvez exécuter la tâche « Run Pylint » pour exécuter *pylint* sur le projet.

---

**Note :** Une note *pylint* supérieure à 9/10 est requise pour fusionner une demande d'extraction.

---

### Règles de codage spécifiques

En plus des directives de codage génériques, nous avons les directives spécifiques suivantes :

- Écrire des docstrings pour toutes les classes, méthodes et fonctions. Les docstrings doivent suivre le [style Google](#).
- Ajouter des annotations de typage pour toutes les fonctions et méthodes. Les annotations doivent utiliser la syntaxe future (`from __future__ import annotations`)
- Essayez de garder le code aussi simple que possible. Si vous devez écrire un morceau de code complexe, essayez de le diviser en plusieurs fonctions ou classes.
- Ajouter autant de commentaires que possible. Le code doit être auto-explicatif, mais il est toujours utile d'ajouter des commentaires pour expliquer l'idée générale du code, ou pour expliquer certaines parties délicates.
- N'utilisez pas d'instructions `from module import *`, même dans le module `__init__` d'un package.
- Évitez d'utiliser des mixins (héritage multiple) si possible. Il est souvent possible d'utiliser la composition au lieu de l'héritage.
- Évitez d'utiliser les méthodes `__getattr__` et `__setattr__`. Ils sont souvent utilisés pour implémenter une initialisation paresseuse, mais cela peut être fait de manière plus explicite.

### 4.4.3 Workflow Git

Ce document décrit le workflow Git utilisé dans le projet DataLab, basé sur une branche `main`, une branche `develop` et des branches spécifiques aux fonctionnalités. Il définit également comment les correctifs de bogues sont gérés.

---

**Note :** Ce workflow est une version simplifiée du [Gitflow Workflow](#). Il a été adapté pour répondre aux besoins du projet DataLab à l'étape actuelle de développement. À l'avenir, nous pourrions envisager d'adopter un workflow plus complexe, par exemple en ajoutant des branches de publication.

---

#### Modèle de branches

##### Branches principales

- `main` : Représente la version stable et prête pour la production du projet.
- `develop` : Utilisé pour le développement continu et l'intégration de nouvelles fonctionnalités.

##### Branches de fonctionnalités

- `develop/feature_name` : Utilisé pour le développement de nouvelles fonctionnalités.
  - Créé à partir de `develop`.
  - Fusionné de nouveau dans `develop` une fois terminé.
  - Supprimé après la fusion.

##### Branches de correction d'anomalies

- `fix/xxx` : Utilisé pour les corrections générales d'anomalies qui ne sont pas urgentes.
  - Créé à partir de `develop`.
  - Fusionné de nouveau dans `develop` une fois terminé.
  - Supprimé après la fusion.
- `hotfix/xxx` : Utilisé pour les corrections urgentes critiques pour la production.
  - Créé à partir de `main`.
  - Fusionné de nouveau dans `main`.
  - La correction est ensuite sélectionnée dans `develop`.
  - Supprimé après la fusion.

---

**Note :** Les correctifs urgents (correctifs de haute priorité) seront intégrés dans la prochaine version de maintenance (X.Y.Z -> Z+1), tandis que les correctifs (correctifs de basse priorité) seront intégrés dans la prochaine version de fonctionnalité (X.Y -> Y+1).

---

## Workflow pour les nouvelles fonctionnalités

1. Créer une nouvelle branche de fonctionnalité à partir de develop :

```
git checkout develop
git checkout -b develop/feature_name
```

2. Développer la fonctionnalité et valider les modifications.
3. Fusionner la branche de fonctionnalité de nouveau dans develop :

```
git checkout develop
git merge --no-ff develop/feature_name
```

4. Supprimer la branche de fonctionnalité :

```
git branch -d develop/feature_name
```

**Avertissement :** Ne pas laisser les branches de fonctionnalités non fusionnées trop longtemps. Les réorganiser régulièrement sur develop pour minimiser les conflits.

## Workflow pour les corrections d'anomalies régulières

1. Créer une branche de correction d'anomalie à partir de develop :

```
git checkout develop
git checkout -b fix/bug_description
```

2. Appliquer la correction et valider les modifications.
3. Fusionner la branche de correction de nouveau dans develop :

```
git checkout develop
git merge --no-ff fix/bug_description
```

4. Supprimer la branche de correction :

```
git branch -d fix/bug_description
```

**Avertissement :** Ne pas créer une branche `fix/xxx` à partir d'une branche `develop/feature_name`. Toujours créer une branche à partir de `develop` pour garantir que les corrections sont correctement propagées.

# *Incorrect:*

```
git checkout develop/feature_name
git checkout -b fix/wrong_branch
```

# *Correct:*

```
git checkout develop
git checkout -b fix/correct_branch
```

## Workflow pour les correctifs urgents

1. Créer une branche de correctif urgent à partir de main :

```
git checkout main
git checkout -b hotfix/critical_bug
```

2. Appliquer la correction et valider les modifications.
3. Fusionner le correctif de nouveau dans main :

```
git checkout main
git merge --no-ff hotfix/critical_bug
```

4. Faire un cherry-pick du correctif dans develop :

```
git checkout develop
git cherry-pick <commit_hash>
```

5. Supprimer la branche de correctif urgent :

```
git branch -d hotfix/critical_bug
```

**Avertissement :** Ne pas fusionner `fix/xxx` ou `hotfix/xxx` directement dans `main` sans suivre le workflow. Assurez-vous que les correctifs urgents ont fait l'objet de cherry-pick dans `develop` pour éviter de perdre des correctifs dans les futures versions.

## Bonnes pratiques

- Réorganiser régulièrement les branches de fonctionnalités sur `develop` pour rester à jour :

```
git checkout develop/feature_name
git rebase develop
```

- Éviter les branches de longue durée pour minimiser les conflits de fusion.
- S'assurer que les corrections d'anomalies dans `main` sont **font toujours l'objet d'un cherry-pick** dans `develop`.
- Différencier clairement entre `fix/xxx` (correctifs non urgents) et `hotfix/xxx` (correctifs critiques pour la production).

## Conclusion

Ce workflow garantit un processus de développement structuré mais flexible tout en maintenant `main` stable et `develop` toujours à jour avec les dernières modifications.

Il garantit également que les corrections d'anomalies sont correctement gérées et propagées entre les branches.

#### 4.4.4 Mise en place de l'environnement de développement

Démarrer le développement dans le cadre du projet DataLab est facile.

Voici ce dont vous aurez besoin :

1. Un environnement de développement intégré (IDE) pour Python. Nous recommandons [Spyder](#) ou [Visual Studio Code](#), mais tout IDE fera l'affaire.
2. Une distribution Python. Nous recommandons [WinPython](#), sur Windows, ou [Anaconda](#), sur Linux ou Mac. Mais, encore une fois, n'importe quelle distribution Python fera l'affaire.
3. Une structure de projet propre (voir ci-dessous).
4. Des données de test (voir ci-dessous).
5. Des variables d'environnement (voir ci-dessous).
6. Des logiciels tiers (voir ci-dessous).

#### Environnement de développement

Si vous utilisez [Spyder](#), merci de soutenir la communauté scientifique Python open-source !

Si vous utilisez Visual Studio Code, c'est aussi un excellent choix (pour d'autres raisons). Nous recommandons d'installer les extensions suivantes :

Extension	Description
<a href="#">gettext</a>	Coloration syntaxique Gettext
<a href="#">Pylance</a>	Serveur de langage Python
<a href="#">Python</a>	Extension Python
<a href="#">reStructuredText Syntax highlighting</a>	Coloration syntaxique reStructuredText
<a href="#">Ruff</a>	Linter et formateur de code Python extrêmement rapide
<a href="#">Todo Tree</a>	Arbre de tâches
<a href="#">Insert GUID</a>	Insérer un GUID
<a href="#">XML Tools</a>	Outils XML

#### Environnement Python

DataLab nécessite les éléments suivants :

- Python (p.ex. WinPython)
- Paquets Python supplémentaires

Installation de tous les paquets requis :

```
pip install --upgrade -r dev\requirements.txt
```

Voir [Installation](#) pour plus de détails sur les versions de référence de Python et Qt.

Si vous utilisez [WinPython](#), merci de soutenir la communauté scientifique Python open-source !

Le tableau suivant liste les distributions Python actuellement utilisées officiellement :

Version de Python	Statut	Version de WinPython
3.9	OK	3.9.10.0
3.10	OK	3.10.11.1
3.11	OK	3.11.5.0
3.12	OK	3.12.3.0
3.13	OK	3.13.2.0

Nous recommandons fortement d'utiliser les versions `.dot` de WinPython qui sont légères et peuvent être personnalisées selon vos besoins (en utilisant `pip install -r requirements.txt`).

Nous recommandons également d'utiliser une instance WinPython dédiée pour DataLab.

### Données de test

Les données de test de DataLab sont situées dans différents dossiers, selon leur nature ou leur origine.

Les données requises pour les tests unitaires sont situées dans « `cdl\data\tests` » (données publiques)

Un second dossier `%CDL_DATA%` (facultatif) peut être défini pour des tests supplémentaires qui sont encore en cours de développement (ou pour des données confidentielles).

### Variables d'environnement spécifiques

Activer le mode « debug » (pas de redirection `stdin/stdout` vers la console interne) :

```
@REM Mode DEBUG
set DEBUG=1
```

Générer la documentation PDF nécessite LaTeX. Sur Windows, l'environnement suivant :

```
@REM LaTeX executable must be in Windows PATH, for mathematical equations rendering
@REM Example with MiKTeX :
set PATH=C:\\Apps\\miktex-portable\\texmf\\install\\miktex\\bin\\x64;%PATH%
```

La configuration de Visual Studio Code utilisée dans `launch.json` et `tasks.json` (exemples) :

```
@REM Development environment
set CDL_PYTHONEXE=C:\\python-3.9.10.amd64\\python.exe
@REM Folder containing additional working test data
set CDL_DATA=C:\\Dev\\Projets\\CDL_data
```

Fichier `.env` de Visual Studio Code :

- Ce fichier est utilisé pour définir les variables d'environnement de l'application.
- Il est utilisé pour définir la variable d'environnement `PYTHONPATH` à la racine du projet.
- Cela est nécessaire pour pouvoir importer les modules du projet depuis VS Code.
- Pour créer ce fichier, copiez le fichier `.env.template` en `.env` (et ajoutez éventuellement vos propres chemins).

### Windows installer

The Windows installer is built using [WiX Toolset V4.0.5](#). Using the WiX Toolset requires [.NET SDK V6.0](#) minimum.

You may install .NET SDK using `winget` :

```
winget install Microsoft.DotNet.SDK.8
```

Once .NET SDK is installed, the WiX Toolset can be installed and configured using the following commands :

```
dotnet tool install --global wix --version 4.0.5
wix extension add WixToolset.UI.wixext/4.0.5
```

First, you need to generate the installer script from a generic template :

```
python wix/makewxs.py
```

Building the installer is done using the following command :

```
wix build .\wix\DataLab.wxs -ext WixToolset.UI.wixext
```

## Logiciels tiers

Les logiciels suivants peuvent être nécessaires pour maintenir le projet :

Logiciel	Description
<a href="#">gettext</a>	Traductions
<a href="#">Git</a>	Système de contrôle de version
<a href="#">ImageMagick</a>	Utilitaires de manipulation d'images
<a href="#">Inkscape</a>	Editeur de graphiques vectoriels
<a href="#">MikTeX</a>	Distribution LaTeX sur Windows

## 4.4.5 Feuille de route

### Jalons futurs

#### Fonctionnalités ou améliorations

- Ajouter la prise en charge de l'acquisition de données :
  - Il serait intéressant de pouvoir acquérir des données de diverses sources (par exemple, une caméra, un numériseur, un spectromètre, etc.) directement depuis DataLab
  - Cela permettrait d'utiliser DataLab comme logiciel d'acquisition de données, et de traiter les données acquises immédiatement après
  - Bien qu'il n'y ait actuellement pas de conception pour cette fonctionnalité, elle pourrait être implémentée en créant une nouvelle famille de plugins, et en définissant une API commune pour les plugins d'acquisition de données
  - Une des solutions techniques possibles pourrait être de s'appuyer sur [PyMoDAQ](#), un paquet Python pour l'acquisition de données, qui est déjà compatible avec divers dispositifs matériels - *et si on collaborait avec les développeurs de PyMoDAQ ?*
- Créer une bibliothèque mathématique DataLab :
  - Cette bibliothèque serait un paquet Python, et contiendrait toutes les fonctions et algorithmes mathématiques utilisés dans DataLab : - Une API d'algorithmes de bas niveau opérant sur des tableaux NumPy - Le modèle de données non-GUI de base de DataLab (par exemple, signaux, images) - Une API de calcul de haut niveau opérant sur les objets DataLab (par exemple, signaux, images)
  - Elle serait utilisée par DataLab lui-même, mais pourrait également être utilisée par des logiciels tiers (par exemple, des notebooks Jupyter, Spyder, Visual Studio Code, etc.)
  - Enfin, cette bibliothèque serait un bon moyen de partager les fonctionnalités mathématiques de DataLab avec la communauté scientifique : une collection d'algorithmes et de fonctions bien testés, bien documentés et faciles à utiliser
  - *Remarque* : il est déjà possible d'utiliser les fonctionnalités de traitement de DataLab depuis l'extérieur de DataLab en important le paquet *cdl*, mais ce paquet contient également le code GUI, qui n'est pas toujours nécessaire (par exemple, lors de l'utilisation de DataLab à partir d'un notebook Jupyter). L'idée ici est de créer un nouveau paquet qui ne contiendrait que les fonctionnalités mathématiques de DataLab, sans le code GUI.

**Note :** La bibliothèque mathématique DataLab pourrait être l’occasion de reconsidérer la conception des fonctions de traitement de DataLab. Actuellement, les fonctions de traitement travaillant sur des objets de signaux et d’images reposent sur des objets *guidata.dataset.DataSet* pour les paramètres d’entrée. C’est très pratique pour le développeur car cela permet de créer automatiquement une interface graphique pour les fonctions de traitement, mais ce n’est pas très flexible pour l’utilisateur car cela oblige à instancier un objet *DataSet* avec les bons paramètres avant d’appeler la fonction de traitement (ce qui peut être fastidieux surtout lorsqu’il s’agit de fonctions de traitement simples nécessitant seulement quelques paramètres). Ainsi, il pourrait être intéressant de considérer une conception plus flexible et simple, où les paramètres de traitement seraient passés en tant qu’arguments de mot-clé aux fonctions de traitement. Les objets *DataSet* pourraient être gérés en interne par les fonctions de traitement (par exemple, en appelant la méthode *DataSet.create* avec les arguments de mot-clé passés par l’utilisateur). Cela permettrait de conserver la fonctionnalité de génération automatique de l’interface graphique pour les fonctions de traitement, mais permettrait également d’appeler directement les fonctions de traitement avec des arguments de mot-clé, sans avoir à créer d’objet *DataSet* au préalable.

---

- Créer un plugin Jupyter pour l’analyse de données interactive avec DataLab :
  - Il est déjà possible d’utiliser DataLab à partir d’un notebook Jupyter, grâce aux fonctionnalités de contrôle à distance (voir [Contrôle à distance](#)), mais il serait utile d’avoir un plugin dédié
  - Ce plugin permettrait d’utiliser DataLab comme noyau Jupyter, et d’afficher les résultats numériques de DataLab dans des notebooks Jupyter ou vice versa (par exemple, afficher les résultats de Jupyter dans DataLab)
  - Ce plugin permettrait également d’utiliser les fonctionnalités de traitement de DataLab à partir de notebooks Jupyter
  - Un cas d’utilisation typique pourrait également consister à utiliser DataLab pour manipuler des signaux ou des images de manière efficace, et à utiliser Jupyter pour l’analyse de données personnalisée basée sur des algorithmes spécifiques / faits maison
  - Ce plugin pourrait être implémenté en utilisant l’interface du noyau Jupyter (voir ci-dessus)
- Créer un plugin Spyder pour l’analyse de données interactive connectée à DataLab :
  - Il s’agit exactement du même cas d’utilisation que pour le plugin Jupyter, mais pour Spyder
  - Ce plugin pourrait également être implémenté en utilisant l’interface du noyau Jupyter (voir ci-dessus)
- Ajouter la prise en charge des séries temporelles (voir [Issue #27](#))
- Ajouter une interface de noyau Jupyter à DataLab :
  - Cela permettrait d’utiliser DataLab à partir d’autres logiciels, tels que des notebooks Jupyter, Spyder ou Visual Studio Code
  - Cela permettrait également de partager des données entre DataLab et d’autres logiciels (par exemple, afficher les résultats numériques de DataLab dans des notebooks Jupyter ou vice versa, afficher les résultats de Jupyter dans DataLab, etc.)
  - Après un premier examen rapide, il semble que l’interface du noyau Jupyter ne soit pas simple à implémenter, de sorte qu’il peut ne pas être utile de s’y atteler (la communication entre DataLab et Jupyter est actuellement déjà possible grâce aux fonctionnalités de contrôle à distance)

## Maintenance

- 2024 : passer à gRPC pour le contrôle à distance (au lieu de XML-RPC), s’il y a besoin d’un protocole de communication plus efficace (voir [Issue #18](#))
- 2025 : abandonner le support de PyQt5 (fin de vie : mi-2025), et passer à PyQt6 ; cela devrait être simple, grâce à la couche de compatibilité *qtpy* et au fait que *PlotPyStack* est déjà compatible avec PyQt6)

## Autres tâches

- Créer un modèle de plugin DataLab (voir [Issue #26](#))

## Jalons passés

### DataLab 0.11

- Ajouter une fonctionnalité de glisser-déposer aux panneaux de signaux et d'images, pour permettre de réorganiser les signaux et les images (voir [Issue #17](#))
- Ajouter des boutons « Monter » et « Descendre » aux panneaux de signaux et d'images, pour permettre de réorganiser les signaux et les images (voir [Issue #22](#))
- Ajouter des fonctionnalités de convolution 1D, d'interpolation, de rééchantillonnage et d'élimination des tendances

### DataLab 0.10

- Développer un plugin DataLab très simple pour démontrer le système de plugins
- Sérialiser les styles de courbes et d'images dans les fichiers HDF5
- Ajouter une option globale « Rafraîchissement automatique », pour pouvoir désactiver le rafraîchissement automatique de la fenêtre principale lors de l'exécution de plusieurs étapes de traitement, améliorant ainsi les performances
- Améliorer la lisibilité des courbes (par exemple, éviter les lignes en pointillés, utiliser des couleurs contrastées et utiliser l'anti-crênelage)

### DataLab 0.9

- Python 3.11 est la nouvelle référence
- Exécuter les calculs dans un processus séparé :
  - Exécuter un « serveur de calcul » en arrière-plan, dans un autre processus
  - Pour chaque calcul, envoyer les données sérialisées et la fonction de calcul au serveur et attendre le résultat
  - Il est alors possible d'arrêter n'importe quel calcul à tout moment en tuant le processus du serveur et en le redémarrant (éventuellement après avoir incrémenté le numéro de port de communication)
- Optimiser les performances d'affichage des images
- Ajouter une boîte de dialogue de préférences
- Ajouter de nouvelles fonctionnalités de traitement d'images : débruitage, ...
- Résultats du traitement d'images : ajout de la prise en charge des formes polygonales (par exemple, pour la détection des contours)
- Nouveau système de plugins : API pour les extensions tierces
  - Objectif n°1 : un plugin doit être gérable à l'aide d'un seul script Python, qui inclut une extension de *ImageProcessor*, *ActionHandler* et une nouvelle prise en charge du format de fichier
  - Objectif n°2 : les plugins doivent simplement être stockés dans un dossier qui est par défaut le répertoire de l'utilisateur (même dossier que le fichier de configuration « .DataLab.ini »)
- Ajouter un système de macro-commandes :
  - Nouvel éditeur Python intégré
  - Scripts utilisant la même API que les scénarios de test applicatifs de haut niveau
  - Prise en charge de l'enregistrement de macro
- Ajouter un serveur xmlrpc pour permettre le contrôle à distance de DataLab :
  - Contrôle des principales fonctionnalités de DataLab (ouvrir un signal ou une image, ouvrir un fichier HDF5, etc.) et des fonctionnalités de traitement (exécuter un calcul, etc.)
  - Prendre le contrôle de DataLab à partir d'un logiciel tiers

- Exécuter des calculs interactifs à partir d'un IDE (par exemple Spyder ou Visual Studio Code)

#### 4.4.6 Historique des modifications

Voir la page de la [feuille de route](<https://datalab-platform.com/fr/contributing/roadmap.html>) de DataLab pour les jalons futurs et passés.

##### DataLab Version 0.18.2

Informations générales :

- Python 3.13 is now supported, since the availability of the scikit-image V0.25 (see [Issue #104](#) - Python 3.13 : `KeyError: 'area_bbox'`)

Enhancements :

- Added new « Keep results after computation » option in « Processing » section :
  - Before this change, when applying a processing feature (e.g. a filter, a threshold, etc.) on a signal or an image, the analysis results were removed from the object
  - This new option allows to keep the analysis results after applying a processing feature on a signal or an image. Even if the analysis results are not updated, they might be relevant in some use cases (e.g. when using the 2D peak detection feature on an image, and then applying a filter on the image, or summing two images, etc.)

Corrections de bugs :

- Fixed [Issue #138](#) - Image colormaps were no longer stored in metadata (and serialized in HDF5 files) since PlotPy v2.6.3 (this commit, specifically : [PlotPyStack/PlotPy@a37af8a](#))
- Fixed [Issue #137](#) - Arithmetic operations and signal interpolation : dialog box with parameters is not displayed
- Fixed [Issue #136](#) - When processing a signal or an image, the analysis result is kept from original object
  - Before this fix, when processing a signal or an image (e.g. when applying a filter, a threshold, etc.), the analysis result was kept from the original object, and was not updated with the new data. Thus the analysis result was not meaningful anymore, and was misleading the user.
  - This is now fixed : the analysis result is now removed when processing a signal or an image. However it is not recalculated automatically, because there is no way to know which analysis result should be recalculated (e.g. if the user has applied a filter, should the FWHM be recalculated ?) - besides, the current implementation of the analysis features does not allow to recalculate the analysis results automatically when the data is modified. The user has to recalculate the analysis results manually if needed.
- Fixed [Issue #132](#) - Plot analysis results : « One curve per result title » mode ignores ROIs
  - Before this fix, the « One curve per result title » mode was ignoring ROIs, and was plotting the selected result for all objects (signals or images) without taking into account the ROI defined on the objects
  - This is now fixed : the « One curve per result title » mode now takes into account the ROI defined on the objects, and plots the selected result for each object (signal or image) and for each ROI defined on the object
- Fixed [Issue #128](#) - Support long object titles in Signal and Image panels
- Fixed [Issue #133](#) - Remove specific analysis results from metadata clipboard during copy operation
- Fixed [Issue #135](#) - Allow to edit ROI on multiple signals or images at once
  - Before this fix, the ROI editor was disabled when multiple signals or images were selected
  - This is now fixed : the ROI editor is now enabled when multiple signals or images are selected, and the ROI is applied to all selected signals or images (only the ROI of the first selected signal or image is taken into account)
  - This new behavior is consistent with the ROI extraction feature, which allows to extract the ROI on multiple signals or images at once, based on the ROI defined on the first selected signal or image
- Image ROI features :
  - Fixed [Issue #120](#) - ROI extraction on multiple images : defined ROI should not be saved in the first selected object. The design choice is to save the defined ROI neither in the first nor in any of the selected objects : the ROI is only used for the extraction, and is not saved in any object

- Fixed [Issue #121](#) - `AttributeError` when extracting multiple ROIs on a single image, if more than one image is selected
- Fixed [Issue #122](#) - Image masks are not refreshed when removing metadata except for the active image
- Fixed [Issue #123](#) - Image masks are not refreshed when pasting metadata on multiple images, except for the last image
- Text and CSV files :
  - Enhance text file reading by detecting data headers (using a list of typical headers from scientific instruments) and by allowing to skip the header when reading the file
  - Ignore encoding errors when reading files in both open feature and import wizard, hence allowing to read files with special characters without raising an exception
  - Fixed [Issue #124](#) - Text files : support locale decimal separator (different than .)
- Signal analysis features : fixed duplicate results when no ROI is defined
- Fixed [Issue #113](#) - Call to `RemoteClient.open_h5_files` (and `import_h5_file`) fails without passing the optional arguments
- Fixed [Issue #116](#) - `KeyError` exception when trying to remove a group after opening an HDF5 file

### DataLab Version 0.18.1

#### Enhancements :

- FWHM computation now raises an exception when less than two points are found with zero-crossing method
- Improved result validation for array-like results by checking the data type of the result

#### Corrections de bugs :

- Fixed [Issue #106](#) - Analysis : coordinate shifted results on images with ROIs and shifted origin
- Fixed [Issue #107](#) - Wrong indices when extracting a profile from an image with a ROI
- Fixed [Issue #111](#) - Proxy `add_object` method does not support signal/image metadata (e.g. ROI)
- Test data plugin / « Create 2D noisy gauss image » : fixed amplitude calculation in `cdl.tests.data.create_2d_random` for non-integer data types

#### Documentation :

- Fixed path separators in plugin directory documentation
- Corrected left and right area descriptions in workspace documentation
- Updated Google style link in contributing guidelines
- Fixed various French translations in the documentation

### DataLab Version 0.18.0

#### Informations générales :

- PlotPy v2.7 est requis pour cette version.
- Prise en charge de Python 3.8 abandonnée.
- Python 3.13 n'est pas encore pris en charge, car certaines dépendances ne sont pas compatibles avec cette version.

#### Nouvelles fonctionnalités et améliorations :

- Nouvelle fonctionnalité de mode d'opération :
  - Ajout de la fonctionnalité « Mode d'opération » à l'onglet « Traitement » de la boîte de dialogue « Paramètres »
  - Cette fonctionnalité permet de choisir entre les modes d'opération « single » et « pairwise » pour toutes les opérations de base (addition, soustraction, multiplication, division, etc.) :
    - Mode « single » : mode d'opérande unique (mode par défaut : l'opération est effectuée sur chaque objet indépendamment)
    - Mode « pairwise » : mode d'opérande par paire (l'opération est effectuée sur chaque paire d'objets)
  - Cela s'applique à la fois aux signaux et aux images, et aux calculs prenant  $N$  entrées
  - Les calculs prenant  $N$  entrées sont ceux où :
    - $N(\geq 2)$  objets en entrée donnent  $N$  objets en sortie
    - $N(\geq 1)$  objet(s) + 1 objet en entrée donnent  $N$  objets en sortie

- Nouvelles fonctionnalités de ROI (Région d'intérêt) :
  - Nouvelle fonctionnalité de ROI polygonale
  - Refonte complète des interfaces utilisateur de l'éditeur de ROI, améliorant l'ergonomie et la cohérence avec le reste de l'application
  - Refactorisation interne majeure du système de ROI pour le rendre plus robuste (plus de tests) et plus facile à maintenir
- Implémentation de l'[Issue #102](<https://github.com/DataLab-Platform/DataLab/issues/102>) - Lancer DataLab en utilisant *datalab* au lieu de *cdl*. Notez que la commande *cdl* est toujours disponible pour la compatibilité ascendante.
- Implémentation de l'[Issue #101](<https://github.com/DataLab-Platform/DataLab/issues/101>) - Configuration : définir l'interpolation d'image par défaut sur l'anti-crénelage (5 au lieu de 0 pour le plus proche). Ce changement est motivé par le fait qu'une amélioration des performances a été apportée dans PlotPy v2.7 sur Windows, ce qui permet d'utiliser l'interpolation par anti-crénelage par défaut sans impact significatif sur les performances.
- Implémentation de l'[Issue #100](<https://github.com/DataLab-Platform/DataLab/issues/100>) - Utiliser le même programme d'installation et exécutable sur Windows 7 SP1, 8, 10, 11. Avant ce changement, un programme d'installation spécifique était requis pour Windows 7 SP1, en raison du fait que Python 3.9 et les versions ultérieures ne sont pas prises en charge sur cette plateforme. Une solution de contournement a été mise en œuvre pour faire fonctionner DataLab sur Windows 7 SP1 avec Python 3.9.

Corrections de bugs :

- Correction de l'[Issue #103](<https://github.com/DataLab-Platform/DataLab/issues/103>) - *proxy.add\_annotations\_from\_items* : la couleur de la forme de cercle semble être ignorée.

## DataLab Version 0.17.1

PlotPy v2.6.2 est requis pour cette version.

Nouvelles fonctionnalités et améliorations :

- Vue Image :
  - Avant cette version, lors de la sélection d'un grand nombre d'images (par exemple lors de la sélection d'un groupe d'images), l'application était très lente car toutes les images étaient affichées dans la vue image, même si elles étaient toutes superposées sur la même image
  - La solution de contournement était d'activer l'option « Afficher uniquement le premier »
  - Maintenant, pour améliorer les performances, si plusieurs images sont sélectionnées, seule la dernière image de la sélection est affichée dans la vue image si cette dernière image n'a pas de transparence et si les autres images sont complètement recouvertes par cette dernière image
- Clarification : l'action « Afficher uniquement le premier » a été renommée en « Afficher uniquement le premier objet », et une nouvelle icône a été ajoutée à l'action
- API : ajout des propriétés *width* et *height* à la classe *ImageObj* (retourne la largeur et la hauteur de l'image en unités physiques)
- Lanceur Windows « start.pyw » : écriture d'un fichier journal « *datalab\_error.log* » lorsqu'une exception se produit au démarrage

Corrections de bugs :

- Changer le thème de couleur met maintenant à jour correctement tous les composants de l'interface utilisateur de DataLab sans avoir besoin de redémarrer l'application

Autres changements :

- OpenCV est désormais une dépendance facultative :
  - Ce changement est motivé par le fait que le paquet conda OpenCV n'est pas maintenu sur Windows (au moins), ce qui entraîne une erreur lors de l'installation de DataLab avec conda
  - Lorsque OpenCV n'est pas installé, seule la fonctionnalité « Détection de blob OpenCV » ne fonctionnera pas, et un message d'avertissement sera affiché lors de la tentative d'utilisation de cette fonctionnalité

## DataLab Version 0.17.0

PlotPy v2.6 est requis pour cette version.

Nouvelles fonctionnalités et améliorations :

- Le menu « Calcul » a été renommé en « Analyse » pour les panneaux Signal et Image, pour mieux refléter la nature des fonctionnalités de ce menu
- Les régions d'intérêt (ROIs) sont désormais prises en compte partout dans l'application là où cela a du sens, et pas seulement pour les anciennes fonctionnalités du menu « Calcul » (maintenant « Analyse »). Cela clôture l'[Issue #93](<https://github.com/DataLab-Platform/DataLab/issues/93>). Si un signal ou une image a une ROI définie :
  - Les opérations sont effectuées uniquement sur la ROI (sauf si l'opération modifie la forme des données, ou la taille des pixels pour les images)
  - Les fonctionnalités de traitement sont effectuées uniquement sur la ROI (si le type de données de l'objet de destination est compatible avec le type de données de l'objet source, ce qui exclut le seuillage, par exemple)
  - Les fonctionnalités d'analyse sont effectuées uniquement sur la ROI, comme avant
- En conséquence du point précédent, et pour plus de clarté :
  - Les fonctionnalités « Modifier les régions d'intérêt » et « Supprimer toutes les régions d'intérêt » ont été déplacées du vieux menu « Calcul » (maintenant « Analyse ») vers le menu « Édition » où se trouvent toutes les fonctionnalités liées aux métadonnées
  - L'action « Modifier les régions d'intérêt » a été ajoutée aux barres d'outils verticales de la Vue Signal et Image (en deuxième position, après l'action « Voir dans une nouvelle fenêtre »)
- Suite à la correction des problèmes de conversion du type de données des images avec les opérations de base, une nouvelle fonctionnalité « Opération arithmétique » a été ajoutée au menu « Opérations » pour les panneaux Signal et Image. Cette fonctionnalité permet d'effectuer des opérations linéaires sur les signaux et les images, avec les opérations suivantes :
  - Addition :  $\text{obj3} = (\text{obj1} + \text{obj2}) * a + b$
  - Soustraction :  $\text{obj3} = (\text{obj1} - \text{obj2}) * a + b$
  - Multiplication :  $\text{obj3} = (\text{obj1} * \text{obj2}) * a + b$
  - Division :  $\text{obj3} = (\text{obj1} / \text{obj2}) * a + b$
- Amélioration de la gestion de la taille des boîtes de dialogue « Voir dans une nouvelle fenêtre » et « Éditeur de ROI » : la taille par défaut ne sera pas plus grande que la taille de la fenêtre principale de DataLab
- Éditeur de ROI :
  - Ajout de barres d'outils pour les éditeurs de ROI Signal et Image, pour permettre de zoomer et dézoomer, et de réinitialiser facilement le niveau de zoom
  - Réorganisation des boutons dans la boîte de dialogue de l'éditeur de ROI pour une meilleure ergonomie et une cohérence avec l'éditeur d'annotations (boîte de dialogue « Voir dans une nouvelle fenêtre »)
- Application de thème de couleur :
  - Ajout de la prise en charge du thème de couleur (auto, clair, foncé) dans la boîte de dialogue « Paramètres »
  - Le thème de couleur est appliqué sans redémarrer l'application

Corrections de bugs :

- Extraction de profil d'intensité / Profil de segment :
  - Lors de l'extraction d'un profil sur une image avec une ROI définie, la fonction PlotPy associée affiche un message d'avertissement ("UserWarning : Attention : conversion d'un élément masqué en nan.") mais le profil est correctement extrait et affiché, avec des valeurs NaN là où la ROI n'est pas définie.
  - Les valeurs NaN sont maintenant supprimées du profil avant de le tracer
- Les fonctionnalités de traitement simples avec une correspondance un-à-un avec une fonction Python (par exemple *numpy.absolute*, *numpy.log10*, etc.) et sans paramètres : correction du titre de l'objet résultat qui se terminait systématiquement par « | » (le caractère qui précède généralement la liste des paramètres)
- Filtrage de Butterworth : correction de la valeur par défaut et de la plage de valeurs valides de la fréquence de coupure
- Toutes les actions liées à la visualisation sont désormais regroupées dans la barre d'outils verticale du panneau de visualisation
  - Lors du démarrage de DataLab avec le panneau Signal actif, le passage à la Vue Image affichait les actions

- « Voir dans une nouvelle fenêtre » ou « Modifier les régions d'intérêt » activées dans la barre d'outils verticale, même si aucune image n'était affichée dans la Vue Image
- La barre d'outils verticale de la Vue Image est maintenant correctement mise à jour au démarrage
- Voir dans une nouvelle fenêtre : les outils de coupe transversale (profils d'intensité) restaient désactivés à moins que l'utilisateur ne sélectionne une image via la liste des éléments - c'est maintenant corrigé
- Vue Image : le bouton de la barre d'outils « Afficher le panneau de contraste » n'était pas activé au démarrage, et n'était activé que lorsqu'au moins une image était affichée dans la Vue Image - il est maintenant toujours activé, comme prévu
- Conversion du type de données des images :
  - Précédemment, la fonction de conversion de type de données était commune aux fonctionnalités de traitement de signal et d'image, c'est-à-dire une simple conversion du type de données en utilisant la méthode *astype* de NumPy
  - Cela n'était pas suffisant pour les fonctionnalités de traitement d'image, en particulier pour les images avec des données de type entier, car même si le résultat était correct d'un point de vue numérique, un débordement ou un dépassement pouvait légitimement être considéré comme un bug d'un point de vue mathématique
  - La fonction de conversion du type de données des images repose désormais sur la fonction interne *clip\_astype*, qui recadre les données dans la plage valide du type de données cible avant de les convertir (dans le cas des images de type entier)
- Problèmes d'extraction de ROI d'image :
  - Plusieurs régressions ont été introduites dans la version 0.16.0 :
    - L'extraction d'une seule ROI circulaire ne fonctionnait pas comme prévu (une ROI rectangulaire était extraite, avec des coordonnées inattendues)
    - L'extraction de plusieurs ROI circulaires conduisait à une extraction de ROI rectangulaire
    - L'extraction de plusieurs ROI ne recadrait plus l'image à la boîte englobante globale des ROIs
  - Ces problèmes sont maintenant corrigés, et des tests unitaires ont été ajoutés pour éviter les régressions :
    - Un algorithme de test indépendant a été implémenté pour vérifier la correction de l'extraction de ROI dans tous les cas mentionnés ci-dessus
    - Les tests couvrent à la fois l'extraction de ROI unique et multiple, avec des ROIs circulaires et rectangulaires
- Problèmes de débordement et de dépassement dans certaines opérations sur des images de type entier :
  - Lors du traitement d'images de type entier, certaines fonctionnalités provoquaient des problèmes de débordement ou de dépassement, entraînant des résultats inattendus (résultats corrects d'un point de vue numérique, mais pas d'un point de vue mathématique)
  - Ce problème ne concernait que les opérations de base (addition, soustraction, multiplication, division et opérations constantes) - toutes les autres fonctionnalités fonctionnaient déjà comme prévu
  - This is now fixed as result output are now floating point images
  - Des tests unitaires ont été ajoutés pour éviter les régressions pour toutes ces opérations

## DataLab Version 0.16.4

Cette version concerne une maintenance mineure.

Corrections de bugs :

- PlotPy v2.4.1 ou ultérieur est requis pour corriger les problèmes suivants liés à la fonction d'ajustement de contraste :
  - Une régression a été introduite dans une version antérieure de PlotPy : l'histogramme des niveaux n'était plus supprimé du panneau d'ajustement de contraste lorsque l'image associée était supprimée du graphique
  - Cela est maintenant corrigé : lorsque une image est supprimée, l'histogramme est également supprimé et le panneau de contraste est rafraîchi (ce qui n'était pas le cas même avant la régression)
- Ignorer *AssertionError* dans *config\_unit\_test.py* lors de l'exécution de la suite de tests sur WSL

Documentation :

- Correction de la référence de classe dans la documentation de *Wrap11Func*

### DataLab Version 0.16.3

Corrections de bugs :

- Correction de l'[Issue #84](https://github.com/DataLab-Platform/DataLab/issues/84) - Problèmes de construction avec V0.16.1 : conflit de nom *signal*, ...
  - Ce problème devait être corrigé dans la version 0.16.2, mais la correction n'était pas suffisante
  - Merci à [@rolandmas](https://github.com/rolandmas) pour avoir signalé le problème et pour l'aide apportée dans l'investigation du problème et le test de la correction
- Correction de l'[Issue #85](https://github.com/DataLab-Platform/DataLab/issues/85) - Les chemins des données de test peuvent être ajoutés plusieurs fois à *cdl.utils.tests.TST\_PATH*
  - Ce problème est lié à l'[Issue #84](https://github.com/DataLab-Platform/DataLab/issues/84)
  - Ajouter les chemins des données de test plusieurs fois à *cdl.utils.tests.TST\_PATH* entraînait le chargement multiple des données de test, ce qui a conduit à l'échec de certains tests (une solution de contournement simple a été ajoutée à V0.16.2 : ce problème est maintenant corrigé)
  - Merci encore à [@rolandmas](https://github.com/rolandmas) pour avoir signalé le problème dans le contexte de l'emballage Debian
- Correction de l'[Issue #86](https://github.com/DataLab-Platform/DataLab/issues/86) - La moyenne de N images entières dépasse le type de données
- Correction de l'[Issue #87](https://github.com/DataLab-Platform/DataLab/issues/87) - Extraction du profil moyen de l'image : *AttributeError* lors de la tentative de modification des paramètres du profil
- Correction de l'[Issue #88](https://github.com/DataLab-Platform/DataLab/issues/88) - Profil de segment d'image : inversion des coordonnées des points

### DataLab Version 0.16.2

Cette version nécessite PlotPy v2.4.0 ou ultérieure, qui apporte les corrections de bugs et les nouvelles fonctionnalités suivantes :

- Nouvelles fonctionnalités d'ajustement de contraste et corrections de bugs :
  - Nouvelle disposition : la barre d'outils verticale (qui était contrainte dans une petite zone sur le côté droit du panneau) est désormais une barre d'outils horizontale en haut du panneau, à côté du titre
  - Nouveau bouton « Définir l'échelle » : permet à l'utilisateur de définir manuellement les valeurs minimale et maximale de la plage de l'histogramme
  - Correction des problèmes de mise à jour de l'histogramme lorsque aucune image n'était actuellement sélectionnée (même si une image était affichée et sélectionnée auparavant)
  - La plage de l'histogramme n'était pas mise à jour lorsque la valeur minimale ou maximale était définie à l'aide des boutons « Valeur minimale » ou « Valeur maximale » (qui ont été renommés en « Min. » et « Max. » dans cette version)
  - La plage de l'histogramme n'était pas mise à jour lorsque le bouton « Définir la plage complète » était cliqué, ou lorsque la plage de LUT était modifiée à l'aide du formulaire « Échelles / Plage de LUT » dans le groupe « Propriétés »
- Menu contextuel de la Vue Image : nouvelle fonctionnalité « Inverser l'axe X »

Nouvelles fonctionnalités mineures et améliorations :

- Types de fichiers image :
  - Ajout de la prise en charge native de la lecture des fichiers image .SPE, .GEL, .NDPI et .REC
  - Ajout de la prise en charge de tout format de fichier pris en charge par *imageio* via le fichier de configuration (l'entrée *imageio\_formats* peut être personnalisée pour compléter la liste par défaut des formats pris en charge : voir la [documentation](https://datalab-platform.com/fr/features/image/menu\_file.html#open-image) pour plus de détails)

Corrections de bugs :

- Analyse de Fourier des images :
  - Correction de l'échelle logarithmique pour le spectre de magnitude (calcul en dB au lieu du logarithme naturel)
  - Correction du calcul de la DSP avec une échelle logarithmique (calcul en dB au lieu du logarithme naturel)

- Mise à jour de la documentation pour mentionner explicitement que l'échelle logarithmique est en dB
- Correction de l'[Issue #82](<https://github.com/DataLab-Platform/DataLab/issues/82>) - Les macros ne sont pas renommées dans DataLab après les avoir exportées vers des scripts Python
- L'objet *ResultProperties* peut désormais être ajouté aux métadonnées de *SignalObj* ou *ImageObj* même en dehors d'une boucle d'événements Qt (car l'élément d'étiquette n'est plus créé immédiatement)
- La barre de progression est désormais automatiquement fermée en cas d'erreur lors d'une opération longue (par exemple, lors de l'ouverture d'un fichier)
- Soustraction, division, ... : la boîte de dialogue pour la sélection du second opérande permettait de sélectionner un groupe (seul un signal ou une image devrait être sélectionné)
- Lorsqu'une opération implique un objet (signal ou image) ayant un numéro d'ordre supérieur à l'objet actuel (par exemple, lors de la soustraction d'une image avec une image d'un groupe en dessous de l'image actuelle), le titre de l'objet résultant fait désormais correctement référence aux numéros d'ordre des objets impliqués dans l'opération (par exemple, pour continuer avec l'exemple de soustraction mentionné ci-dessus, le titre de l'objet résultant faisait précédemment référence au numéro d'ordre avant l'insertion de l'image résultante)
- Ajout de la prise en charge d'un dossier de données de test supplémentaire grâce à la variable d'environnement *CDL\_DATA* (utile à des fins de test, et notamment dans le contexte de la préparation du paquet Debian de DataLab)

## DataLab Version 0.16.1

Depuis la version 0.16.0, de nombreuses fonctions de validation ont été ajoutées à la suite de tests. Le pourcentage de fonctions de calcul validées est passé de 37 % à 84 % dans cette version.

NumPy 2.0 support has been added with this release.

Nouvelles fonctionnalités mineures et améliorations :

- Filtres moyenne mobile et médiane pour les signaux et les images :
  - Ajout du paramètre « Mode » pour choisir le mode du filtre (par exemple « reflect », « constant », « nearest », « mirror », « wrap »)
  - Le mode par défaut est « reflect » pour la moyenne mobile et « nearest » pour la médiane mobile
  - Cela permet de gérer les effets de bord lors du filtrage des signaux et des images

Corrections de bugs :

- Correction de la détection de bord de Canny pour renvoyer une image binaire en *uint8* au lieu de *bool* (par soucis de cohérence avec les autres fonctionnalités de traitement d'image)
- Correction de la normalisation d'image : la borne inférieure était mal définie pour la méthode *maximum*
- Correction de l'erreur *ValueError* lors du calcul de la DSP avec une échelle logarithmique
- Correction de l'algorithme de dérivation d'un signal : utilisation de *numpy.gradient* au lieu d'une implémentation personnalisée
- Correction de l'obsolescence de *cumtrapz* de SciPy : utilisation de *cumulative\_trapezoid* à la place
- La sélection de courbe affiche désormais les points individuels de la courbe (auparavant, seule la largeur de la ligne de la courbe était élargie)
- Installateur Windows : ajout de la prise en charge des versions instables (par exemple, 0.16.1.dev0), permettant ainsi d'installer facilement la dernière version de développement de DataLab sur Windows
- Correction de l'[Issue #81](<https://github.com/DataLab-Platform/DataLab/issues/81>) - Lors de l'ouverture de fichiers, afficher la boîte de dialogue de progression uniquement si nécessaire
- Correction de l'[Issue #80](<https://github.com/DataLab-Platform/DataLab/issues/80>) - Tracé des résultats : prise en charge de deux cas d'utilisation
  - Les fonctionnalités du menu Analyse » produisent des *résultats* (scalaires) : détection de taches (coordonnées de cercle), détection de pics 2D (coordonnées de points), etc. Selon la fonctionnalité, des tables de résultats sont affichées dans la boîte de dialogue « Résultats », et les résultats sont également stockés dans les métadonnées du signal ou de l'image : chaque ligne de la table de résultats est un résultat individuel, et chaque colonne est une propriété du résultat - certains résultats peuvent ne consister qu'en un seul résultat individuel (par exemple, le centroïde de l'image ou la FWHM de la courbe), tandis que d'autres peuvent consister en plusieurs résultats individuels (par exemple, détection de taches, détection de contours, etc.).

- Avant ce changement, la fonctionnalité « Tracer les résultats » ne prenait en charge que le tracé du premier résultat individuel d'une table de résultats, en fonction de l'index (des objets de signal ou d'image) ou de l'une des colonnes de la table de résultats. Cela n'était pas suffisant pour certains cas d'utilisation, où l'utilisateur souhaitait tracer plusieurs résultats individuels d'une table de résultats.
- A présent, la fonctionnalité « Tracer les résultats » prend en charge deux cas d'utilisation :
  - « Une courbe par titre de résultat » : Tracer le premier résultat individuel d'une table de résultats, comme auparavant
  - « Une courbe par objet (ou ROI) et par titre de résultat » : Tracer tous les résultats individuels d'une table de résultats, en fonction de l'index (des objets de signal ou d'image) ou de l'une des colonnes de la table de résultats
- La sélection du cas d'utilisation se fait dans la boîte de dialogue « Tracer les résultats »
- Le cas d'utilisation par défaut est « Une courbe par titre de résultat » si la table de résultats ne comporte qu'une seule ligne, et « Une courbe par objet (ou ROI) et par titre de résultat » dans les autres cas

## DataLab Version 0.16.0

### Nouvelles fonctionnalités et améliorations :

- Refonte majeure de l'interface utilisateur :
  - La barre de menu et les barres d'outils ont été réorganisées pour rendre l'application plus intuitive et plus facile à utiliser
  - Les fonctionnalités d'opérations et de traitement ont été regroupées dans des sous-menus
  - Toutes les actions liées à la visualisation sont désormais regroupées dans la barre d'outils verticale du panneau de visualisation
  - Clarification de la gestion des « Annotations » (nouveaux boutons, action de la barre d'outils, ...)
- Nouveau processus de validation pour les fonctionnalités de traitement du signal et de l'image :
  - Avant cette version, le processus de validation de DataLab était exclusivement effectué du point de vue du programmeur, en écrivant des tests unitaires et des tests d'intégration, garantissant ainsi que le code fonctionnait comme prévu (c'est-à-dire qu'aucune exception n'était levée et que le comportement était correct)
  - Avec cette version, un nouveau processus de validation a été introduit, du point de vue de l'utilisateur, en ajoutant de nouvelles fonctions de validation (marquées avec le décorateur `@pytest.mark.validation`) dans la suite de tests
  - Une nouvelle section « Validation » dans la documentation explique comment la validation est effectuée et contient une liste de toutes les fonctions de validation avec les statistiques du processus de validation (générées à partir de la suite de tests)
  - Le processus de validation est en cours et sera amélioré dans les versions futures
- Groupe « Propriétés » :
  - Ajout de l'onglet « Échelles », pour afficher et définir les échelles du graphique :
    - X, Y pour les signaux
    - X, Y, Z (plage de LUT) pour les images
- Options d'affichage :
  - Nouvelle option « Afficher uniquement le premier » dans le menu « Affichage », pour afficher uniquement la première courbe (ou image) lorsque plusieurs courbes (ou images) sont affichées dans la vue du graphique
  - Nouvelle étiquette (déplaçable) pour les calculs de FWHM, en plus de l'annotation de segment existante
- Fonctionnalités d'E/S :
  - Ajout de la prise en charge de la lecture et de l'écriture des fichiers .MAT (format MATLAB)
  - Création d'un nouveau groupe lors de l'ouverture d'un fichier contenant plusieurs signaux ou images (par exemple, un fichier CSV avec plusieurs courbes)
  - Ajout de la prise en charge des images binaires
  - Extraction de ROI de signal : ajout d'une nouvelle boîte de dialogue pour éditer manuellement les bornes inférieure et supérieure du ROI après avoir défini le ROI graphiquement

### Nouvelles fonctionnalités de traitement des **Signaux** :

Menu	Sous-menu	Fonctionnalités
Nouveau	Nouveau signal	Exponentielle, impulsion, polynomiale, expérimentale (entrée manuelle)
Opérations		Exponentielle, Racine carrée, Puissance
Opérations	Opérations avec une constante	+, -, *, /
Traitement	Transformation des axes	Inverser l'axe X
Traitement	Ajustement du niveau	Correction d'offset
Traitement	Analyse de Fourier	Spectre de puissance, Spectre de phase, Spectre de magnitude, Densité spectrale de puissance
Traitement	Filtres fréquentiels	Passe-bas, Passe-haut, Passe-bande, Coupure de bande
Traitement		Fenêtrage (Hanning, Hamming, Blackman, Blackman-Harris, Nuttall, Flat-top, ...)
Traitement	Ajustement	Ajustement linéaire, Ajustement sinusoïdal, Ajustement exponentiel, Ajustement CDF
Analyse		LMH (Méthode du zéro-crossing), Valeur X @ min/max, Période/fréquence d'échantillonnage, Paramètres dynamiques (ENOB, SNR, SINAD, THD, SFDR), Largeur de bande - 3dB, Contraste

Nouvelles fonctionnalités de traitement des **Images** :

Menu	Sous-menu	Fonctionnalités
Opérations		Exponentielle
Opérations	Profils d'intensité	Profil le long d'un segment
Opérations	Opérations avec une constante	+, -, *, /
Traitement	Ajustement du niveau	Normalisation, Recadrage, Correction d'offset
Traitement	Analyse de Fourier	Spectre de puissance, Spectre de phase, Spectre de magnitude, Densité spectrale de puissance
Traitement	Seuillage	Paramétrique, ISODATA, Li, Moyenne, Minimum, Otsu, Triangle, Yen

Corrections de bugs :

- Correction d'un problème de performance dû à un rafraîchissement inutile de la vue du graphique lors de l'ajout d'un nouveau signal ou d'une nouvelle image

- Correction de l'[Issue #77](https://github.com/DataLab-Platform/DataLab/issues/77) - Profils d'intensité : impossible d'accepter la boîte de dialogue la deuxième fois
- Correction de l'[Issue #75](https://github.com/DataLab-Platform/DataLab/issues/75) - Afficher dans une nouvelle fenêtre : l'anti-crénelage de la courbe n'est pas activé par défaut
- La visibilité des annotations est désormais correctement sauvegardée et restaurée :
  - Avant cette version, lorsque la visibilité des annotations était modifiée dans la vue du graphique séparée, elle n'était pas sauvegardée et restaurée lors de la réouverture de la vue du graphique
  - Cela a été [corrigé en amont](https://github.com/PlotPyStack/PlotPy/commit/03faaa42e5d6d4016ea8c99334c29d46a5963467) dans PlotPy (v2.3.3)

### DataLab Version 0.15.1

#### Corrections de bugs :

- Correction de l'[Issue #68](https://github.com/DataLab-Platform/DataLab/issues/68) - Chargement lent même pour des graphiques simples :
  - Sur macOS, l'expérience utilisateur était dégradée lors de la manipulation de graphiques simples
  - Cela était dû à la manière dont macOS gère les fenêtres contextuelles, par exemple lors du rafraîchissement de la vue du graphique (barre de progression « Création d'éléments de graphique »), provoquant ainsi un effet de scintillement très gênant et un ralentissement global de l'application
  - Le problème est maintenant résolu en affichant la barre de progression uniquement après un court délai (1s), c'est-à-dire lorsqu'elle est vraiment nécessaire (c'est-à-dire pour les opérations longues)
  - Merci à [@marcel-goldschen-ohm](https://github.com/marcel-goldschen-ohm) pour la description très détaillée du problème et l'aide apportée pour tester la correction
- Correction de l'[Issue #68](https://github.com/DataLab-Platform/DataLab/issues/69) - Les annotations devaient être en lecture seule dans la vue Signal/Image
  - En ce qui concerne les annotations, le comportement actuel de DataLab est le suivant :
    - Les annotations ne sont créées que lors de l'affichage du signal/image dans une fenêtre séparée (double-clic sur l'objet, ou « Affichage » > « Afficher dans une nouvelle fenêtre »)
    - Quand les objets sont affichés dans la « Vue Signal » ou la « Vue Image », les annotations devaient être en lecture seule (c'est-à-dire non déplaçables, ni redimensionnables ou supprimables)
  - Toutefois, certaines annotations étaient toujours supprimables dans la « Vue Signal » et la « Vue Image » : c'est maintenant corrigé
  - Notons que le fait que les annotations ne peuvent pas être créées dans la « Vue Signal » ou la « Vue Image » est une limitation de l'implémentation actuelle, et peut être amélioré dans les versions futures

### DataLab Version 0.15.0

#### Nouvel installateur pour la version autonome sur Windows :

- La version autonome sur Windows est désormais distribuée en tant qu'installateur MSI (au lieu d'un installateur EXE)
- Cela évite la détection de faux positifs de la version autonome comme une menace potentielle par certains logiciels antivirus
- Le programme installera des fichiers et des raccourcis :
  - Pour l'utilisateur actuel, si l'utilisateur n'a pas de privilèges d'administrateur
  - Pour tous les utilisateurs, si l'utilisateur a des privilèges d'administrateur
  - Le répertoire d'installation peut être personnalisé
- L'installateur MSI permet d'intégrer l'installation de DataLab de manière transparente dans le système de déploiement d'une organisation

#### Nouvelles fonctionnalités et améliorations :

- Ajout de la prise en charge des fichiers texte/CSV volumineux :
  - Les fichiers de plus de 1 Go (et avec un nombre raisonnable de lignes) peuvent désormais être importés en tant que signaux ou images sans quitter inopinément l'application ou même la ralentir

- Le fichier est lu par morceaux et, pour les signaux, les données sont rééchantillonnées à un nombre raisonnable de points pour la visualisation
  - Les fichiers volumineux sont pris en charge lors de l'ouverture d'un fichier (ou du glisser-déposer d'un fichier dans le Panneau Signal) et lors de l'importation d'un fichier dans l'Assistant d'importation de texte
  - Nouvelle fonctionnalité de sous-échantillonnage automatique :
    - Ajout de la fonctionnalité « Sous-échantillonnage automatique » aux paramètres de visualisation des signaux (voir la boîte de dialogue « Paramètres »)
    - Cette fonctionnalité permet de rééchantillonner automatiquement les données du signal pour la visualisation lorsque le nombre de points est trop élevé et entraînerait un rendu lent
    - Le facteur de sous-échantillonnage est automatiquement calculé en fonction du nombre maximal de points à afficher configuré
    - Cette fonctionnalité est activée par défaut et peut être désactivée dans les paramètres de visualisation du signal
  - Gestion du format CSV :
    - Amélioration de la prise en charge des fichiers CSV avec une ligne d'en-tête (noms de colonnes)
    - Ajout de la prise en charge des fichiers CSV avec des colonnes vides
  - Gestion des erreurs d'ouverture/enregistrement de fichiers :
    - Les messages d'erreur sont désormais plus explicites lorsque l'ouverture ou l'enregistrement d'un fichier échoue
    - Ajout d'un lien vers le dossier contenant le fichier dans le message d'erreur
  - Ajout de la page « Plugins et fonctionnalités d'entrée/sortie » au Visualiseur d'installation et de configuration (voir le menu « Aide »)
  - Réinitialisation de la configuration de DataLab :
    - Dans certains cas, il peut être utile de réinitialiser le fichier de configuration de DataLab à ses valeurs par défaut (par exemple, lorsque le fichier de configuration est corrompu)
    - Ajout de la nouvelle option de ligne de commande *-reset* pour supprimer le dossier de configuration
    - Ajout du nouveau raccourci du Menu Démarrer « Reset DataLab » à l'installateur Windows
- Corrections de bugs :
- Correction de l'[Issue #64](<https://github.com/DataLab-Platform/DataLab/issues/64>) - L'explorateur HDF5 ne montre pas les ensembles de données de taille 1x1 :
    - Les datasets HDF5 de taille 1x1 n'étaient pas affichés dans l'explorateur HDF5
    - Même si ces données ne devraient pas être considérées comme des signaux ou des images, elles sont désormais affichées dans l'explorateur HDF5 (mais non sélectionnables, c'est-à-dire non importables en tant que signaux ou images)

## **DataLab Version 0.14.2**

Modifications de l'API nécessaires pour corriger la fonctionnalité de chargement de plusieurs signaux :

- Fusion des méthodes *open\_object* et *open\_objects* en *load\_from\_files* dans les classes proxy, la fenêtre principale et les panneaux de données
- Pour des raisons de cohérence : fusion de *save\_object* et *save\_objects* en *save\_to\_files*
- En résumé, ces changements mènent à la situation suivante :
  - *load\_from\_files* : charge une séquence d'objets à partir de plusieurs fichiers
  - *save\_to\_files* : sauvegarde une séquence d'objets dans plusieurs fichiers (pour le moment, il ne prend en charge que la sauvegarde d'un seul objet dans un seul fichier, mais il pourrait être étendu à l'avenir pour prendre en charge la sauvegarde de plusieurs objets dans un seul fichier)

Corrections de bugs :

- Correction de l'[Issue #61](<https://github.com/DataLab-Platform/DataLab/issues/61>) - Assistant d'importation de fichiers texte : plantage de l'application lors de l'importation d'un fichier texte à courbes multiples :
  - Ce problème concerne un cas d'utilisation où le fichier texte contient plusieurs courbes
  - C'est maintenant corrigé et un test automatique a été ajouté pour éviter les régressions

## DataLab Version 0.14.1

Nouveau nom de domaine : [datalab-platform.com](https://datalab-platform.com)

Nouvelles fonctionnalités :

- Ajout de la prise en charge de l'inversion de la palette de couleurs dans la Vue Image :
  - Ajout de la nouvelle entrée « Inverser la palette de couleurs » dans le menu contextuel du graphique, les paramètres de l'image et dans les paramètres par défaut de la vue image
  - Cette fonctionnalité nécessite *PlotPy* v2.3 ou ultérieur
- Explorateur HDF5 :
  - Ajout du bouton « Afficher le tableau » dans le coin des onglets « Groupe » et « Attributs », pour afficher le tableau dans une fenêtre séparée (utile pour copier/coller des données dans d'autres applications, par exemple)
  - Attributs : ajout de la prise en charge de plus de types de données scalaires
- Testabilité et maintenabilité :
  - Les tests unitaires de DataLab utilisent désormais *[pytest]*(https://pytest.org). Cela a nécessité beaucoup de travail pour la transition, en particulier pour réadapter les tests afin qu'ils puissent être exécutés dans le même processus. Par exemple, une attention particulière a été portée à l'isolement des tests, afin qu'ils n'interfèrent pas les uns avec les autres.
  - Ajout de l'intégration continue (CI) avec GitHub Actions
  - Pour cette version, la couverture des tests est de 87%
- Assistant d'importation de fichiers texte :
  - Amélioration drastique des performances de l'aperçu du tableau lors de l'importation de fichiers texte volumineux (plus de barre de progression, et l'aperçu est désormais affiché presque instantanément)

Corrections de bugs :

- Le serveur XML-RPC n'était pas arrêté correctement lors de la fermeture de DataLab
- Correction de problèmes liés aux tests : certains cas limites étaient masqués par l'ancienne suite de tests, et ont été révélés par la transition vers *pytest*. Cela a conduit à des corrections de bugs et à des améliorations du code.
- Sur Linux, lors de l'exécution d'un calcul sur un signal ou une image, et à de rares occasions, le calcul restait bloqué comme s'il s'exécutait indéfiniment. Même si l'interface graphique était toujours réactive, le calcul n'avancait pas et l'utilisateur devait annuler l'opération et la redémarrer. Cela était dû à la méthode de démarrage du processus séparé utilisé pour le calcul (la méthode par défaut était « fork » sur Linux). Ceci est maintenant corrigé en utilisant la méthode « spawn » à la place, qui est la méthode recommandée pour les dernières versions de Python sur Linux lorsque le multithreading est employé.
- Correction de l'[Issue #60](https://github.com/DataLab-Platform/DataLab/issues/60) - *OSError : Invalid HDF5 file [...]* lors de la tentative d'ouverture d'un fichier HDF5 avec une extension autre que « .h5 »
- Région d'intérêt (ROI) d'image : lors de la modification des limites de l'image dans la boîte de dialogue de confirmation, la ROI n'était pas mise à jour en conséquence jusqu'à ce que l'opération soit relancée
- Problèmes d'obsolescence :
  - Correction de l'avertissement d'obsolescence *scipy.ndimage.filters*
  - Correction de l'avertissement d'obsolescence *numpy.fromstring*

## DataLab Version 0.14.0

Nouvelles fonctionnalités :

- Nouvelle fonctionnalité « Histogramme » dans le menu Analyse » :
  - Ajout de la fonctionnalité de calcul d'histogramme pour les signaux et les images
  - L'histogramme est calculé sur les régions d'intérêt (ROI) le cas échéant, ou sur l'ensemble du signal/image si aucune ROI n'est définie
  - Paramètres modifiables : nombre de classes, limites inférieure et supérieure
- Explorateur HDF5 :
  - Amélioration de la disposition de la vue arborescente (plus compacte et lisible)
  - Plusieurs fichiers peuvent désormais être ouverts en même temps, en utilisant la boîte de dialogue de sélection de fichiers

- Ajout d'onglets avec des informations sous l'aperçu graphique :
  - Informations sur le groupe : chemin, aperçu textuel, etc.
  - Informations sur les attributs : nom, valeur
- Ajout de la case à cocher « Afficher uniquement les données prises en charge » : lorsqu'elle est cochée, seules les données prises en charge (signaux et images) sont affichées dans la vue arborescente
- Ajout de la case à cocher « Afficher les valeurs », pour afficher/masquer les valeurs dans la vue arborescente
- Panneau de macro-commandes :
  - Les macro-commandes sont désormais numérotées, à partir de 1, comme les signaux et les images
- API de contrôle à distance (*RemoteProxy* et *LocalProxy*) :
  - *get\_object\_titles* accepte désormais « macro » comme nom de panneau et retourne la liste des titres de macro
  - Nouvelles méthodes *run\_macro*, *stop\_macro* et *import\_macro\_from\_file*

#### Corrections de bugs :

- Version autonome - Intégration dans le menu de démarrage de Windows :
  - Correction du raccourci « Désinstaller » (non cliquable en raison d'un nom générique)
  - Traduction des raccourcis « Parcourir le répertoire d'installation » et « Désinstaller »
- Correction de l'[Issue #55](<https://github.com/DataLab-Platform/DataLab/issues/55>) - Changer les limites de l'image dans la vue d'image n'a aucun effet sur les propriétés de l'objet image associé
- Correction de l'[Issue #56](<https://github.com/DataLab-Platform/DataLab/issues/56>) - Plugin « Données de test » : *AttributeError* : “NoneType” object has no attribute “data” lors de l'annulation de « Créer une image avec des pics »
- Ceci corrige l'[Issue #57](<https://github.com/DataLab-Platform/DataLab/issues/57>) - Les formes résultat cercle et ellipse ne sont pas transformées correctement
- Cycle de couleur et de style de courbe :
  - Avant cette version, ce cycle était géré par le même mécanisme pour le Panneau Signal ou l'Explorateur HDF5, ce qui n'était pas le comportement attendu
  - A présent, le cycle est géré séparément : l'Explorateur HDF5 ou l'Assistant d'importation de texte utilisent toujours la même couleur et le même style pour les courbes, et ils n'interfèrent pas avec le cycle du Panneau Signal

### DataLab Version 0.12.0

#### Clarification de certains libellés des interfaces graphiques :

- Les onglets utilisés pour basculer entre les panneaux de données (signaux et images) et les composants de visualisation (« Panneau de courbes » et « Panneau d'images ») ont été renommés en « Panneau Signal » et « Panneau Image » (au lieu de « Signaux » et « Images »)
- Les composants de visualisation ont été renommés en « Vue Signal » et « Vue Image » (au lieu de « Panneau de courbes » et « Panneau d'images »)
- Les barres d'outils des panneaux de données ont été renommées en « Barre d'outils Signal » et « Barre d'outils Image » (au lieu de « Barre d'outils Traitement du Signal » et « Barre d'outils Traitement d'Image »)
- Améliorations ergonomiques : le « Panneau Signal » et le « Panneau Image » sont désormais affichés sur le côté gauche de la fenêtre principale, et la « Vue Signal » et la « Vue Image » sont affichées sur le côté droit de la fenêtre principale. Cela réduit la distance entre la liste des objets (signaux et images) et les actions associées (barres d'outils et menus), et rend l'interface plus intuitive et plus facile à utiliser

#### Nouvelle fonctionnalité de visite guidée et de démonstration :

- Lors du premier démarrage de DataLab, une visite guidée est désormais affichée à l'utilisateur pour présenter les principales fonctionnalités de l'application
- La visite guidée peut être relancée à tout moment depuis le menu « ? »
- Ajout d'une nouvelle fonctionnalité « Démonstration » dans le menu « ? »

#### Nouvel environnement Binder pour tester DataLab en ligne sans rien installer

#### Documentation :

- De nouveaux tutoriels textuels sont disponibles :
  - Mesure de la taille d'un faisceau laser
  - DataLab et Spyder : un mariage parfait

- Section « Premiers pas » : ajout de plus d'explications et de liens vers les tutoriels
- Nouvelle section « Contribuer » expliquant comment contribuer à DataLab, que vous soyez développeur ou non
- Nouvelle section « Macros » expliquant comment utiliser la fonctionnalité de macro-commandes
- Ajout du bouton « Copier » aux blocs de code dans la documentation

#### Nouvelles fonctionnalités :

- Nouvelle fonctionnalité d'assistant d'importation de fichiers texte :
  - Cette fonctionnalité permet d'importer des fichiers texte en tant que signaux ou images
  - L'utilisateur peut définir la source (presse-papiers ou fichier texte)
  - Ensuite, il est possible de définir le délimiteur, le nombre de lignes à sauter, le type de données de destination, etc.
- Ajout d'un menu dans le coin des onglets « Panneau Signal » et « Panneau Image » pour accéder rapidement aux fonctionnalités les plus utilisées (par exemple « Ajouter », « Supprimer », « Dupliquer », etc.)
- Fonctionnalité d'extraction de profil d'intensité :
  - Ajout d'une interface graphique pour extraire des profils d'intensité des images, pour les profils linéaires et moyennés
  - Les paramètres sont toujours directement modifiables par l'utilisateur (bouton « Modifier les paramètres du profil »)
  - Les paramètres sont désormais stockés d'une extraction de profil à l'autre
- Fonctionnalité de statistiques :
  - Ajout de  $\langle y \rangle / \langle y \rangle$  au tableau de résultats « Statistiques » du signal (en plus de la moyenne, de la médiane, de l'écart-type, etc.)
  - Ajout de *peak-to-peak* au tableau de résultats « Statistiques » du signal et de l'image
- Fonctionnalité d'ajustement de courbe : les résultats de l'ajustement sont désormais stockés dans un dictionnaire dans les métadonnées du signal (au lieu d'être stockés individuellement dans les métadonnées du signal)
- État de la fenêtre :
  - L'état des barres d'outils et des widgets de dock (visibilité, position, etc.) est désormais stocké dans le fichier de configuration et restauré au démarrage (la taille et la position étaient déjà stockées et restaurées)
  - Ceci implémente une partie de [Issue #30](https://github.com/DataLab-Platform/DataLab/issues/30) - Sauvegarder/restaurer la disposition de la fenêtre principale

#### Corrections de bugs :

- Correction de l'[Issue #41](https://github.com/DataLab-Platform/DataLab/issues/41) - Extraction du profil radial : impossible d'entrer les coordonnées du centre définies par l'utilisateur
- Correction de l'[Issue #49](https://github.com/DataLab-Platform/DataLab/issues/49) - Erreur lors de l'ouverture d'un fichier texte (UTF-8 BOM) en tant qu'image
- Correction de l'[Issue #51](https://github.com/DataLab-Platform/DataLab/issues/51) - Dimensions inattendues lors de l'ajout d'une nouvelle ROI sur une image avec des unités X/Y arbitraires (pas des pixels)
- Amélioration de la gestion de la sérialisation du style des items graphiques :
  - Avant cette version, le style des items graphiques était stocké dans les métadonnées du signal/image uniquement lors de la sauvegarde de l'espace de travail dans un fichier HDF5. Ainsi, lors de la modification du style d'un signal/image à partir du bouton « Paramètres » (barre d'outils de la vue), le style n'était pas conservé dans certains cas (par exemple lors de la duplication du signal/image).
  - A présent, le style des items graphiques est stocké dans les métadonnées du signal/image chaque fois que le style est modifié, et est restauré lors du rechargement de l'espace de travail
- Traitement de l'avertissement de conversion *ComplexWarning* lors de l'ajout de régions d'intérêt (ROI) à un signal avec des données complexes

## DataLab Version 0.11.0

### Nouvelles fonctionnalités :

- Les signaux et les images peuvent maintenant être réorganisés dans la vue arborescente :
  - En utilisant les nouvelles actions « Monter » et « Descendre » dans le menu « Édition » (ou en utilisant les boutons de la barre d'outils correspondants) :
  - Ceci corrige l'[Issue #22](<https://github.com/DataLab-Platform/DataLab/issues/22>) - Ajout des actions « monter/descendre » dans le menu « Édition », pour les signaux/images et les groupes
- Les signaux et les images peuvent également être réorganisés par glisser-déposer :
  - Les signaux et les images peuvent être déplacés et déposés dans leur propre panneau pour changer leur ordre
  - Les groupes peuvent également être déplacés et déposés dans leur panneau
  - La fonctionnalité prend également en charge la sélection multiple (en utilisant les modificateurs standard Ctrl et Shift), de sorte que plusieurs signaux/images/groupe peuvent être déplacés à la fois, pas nécessairement avec des positions contiguës
  - Ceci corrige l'[Issue #17](<https://github.com/DataLab-Platform/DataLab/issues/17>) - Ajout de la fonctionnalité de glisser-déposer aux vues arborescentes des signaux/images
- Nouvelles fonctionnalités d'interpolation 1D :
  - Ajout de la fonctionnalité « Interpolation » au menu « Traitement » du panneau de signal
  - Méthodes disponibles : linéaire, spline, quadratique, cubique, barycentrique et PCHIP
  - Merci à [@marcel-goldschen-ohm](<https://github.com/marcel-goldschen-ohm>) pour sa contribution à l'interpolation spline
  - Ceci corrige l'[Issue #20](<https://github.com/DataLab-Platform/DataLab/issues/20>) - Ajout des fonctionnalités d'interpolation 1D
- Nouvelle fonctionnalité de rééchantillonnage 1D :
  - Ajout de la fonctionnalité « Rééchantillonnage » au menu « Traitement » du panneau de signal
  - Mêmes méthodes d'interpolation que pour la fonctionnalité « Interpolation »
  - Possibilité de spécifier le pas de rééchantillonnage ou le nombre de points
  - Ceci corrige l'[Issue #21](<https://github.com/DataLab-Platform/DataLab/issues/21>) - Ajout de la fonctionnalité de rééchantillonnage 1D
- Nouvelle fonctionnalité de convolution 1D :
  - Ajout de la fonctionnalité « Convolution » au menu « Opération » du panneau de signal
  - Ceci corrige l'[Issue #23](<https://github.com/DataLab-Platform/DataLab/issues/23>) - Ajout de la fonctionnalité de convolution 1D
- Nouvelle fonctionnalité de d'élimination de tendance 1D :
  - Ajout de la fonctionnalité « Élimination de tendance » au menu « Traitement » du panneau de signal
  - Méthodes disponibles : linéaire ou constante
  - Ceci corrige l'[Issue #24](<https://github.com/DataLab-Platform/DataLab/issues/24>) - Ajout de la fonctionnalité d'élimination de tendance 1D
- Résultats d'analyse 2D :
  - Avant cette version, les résultats d'analyse 2D tels que les contours, les blobs, etc. étaient stockés dans le dictionnaire de métadonnées de l'image sous forme de coordonnées (x0, y0, x1, y1, ...) même pour les cercles et les ellipses (c'est-à-dire les coordonnées des rectangles englobants).
  - Par souci de commodité, les coordonnées du cercle et de l'ellipse sont désormais stockées dans le dictionnaire de métadonnées de l'image sous la forme (x0, y0, rayon) et (x0, y0, a, b, theta) respectivement.
  - Ces résultats sont également affichés comme tels dans la boîte de dialogue « Résultats » (à la fin du processus de calcul ou en cliquant sur le bouton « Afficher les résultats »).
  - Cette correction corrige l'[Issue #32](<https://github.com/DataLab-Platform/DataLab/issues/32>) - Détection des contours : afficher le cercle (x, y, r) et l'ellipse (x, y, a, b, theta) au lieu de (x0, y0, x1, y1, ...)
- Résultats d'analyse 1D et 2D :
  - En complément de l'amélioration précédente, plus de résultats d'analyse sont désormais affichés dans la boîte de dialogue « Résultats »
  - Cela concerne à la fois les résultats d'analyse 1D (FWHM, ...) et 2D (contours, blobs, ...) :
    - Les résultats de type segment affichent désormais également la longueur (L) et les coordonnées du centre (Xc, Yc)

- Les résultats de type cercle et ellipse affichent désormais également l'aire (A)
- Ajout de l'entrée « Tracer les résultats » dans le menu Analyse » :
  - Cette fonctionnalité permet de tracer les résultats d'analyse (1D ou 2D)
  - Cela crée un nouveau signal avec des axes X et Y correspondant à des paramètres définis par l'utilisateur (par exemple X = index et Y = rayon pour les résultats de cercle)
- Augmentation de la largeur par défaut de la boîte de dialogue de sélection d'objet :
  - La boîte de dialogue de sélection d'objet est désormais plus large par défaut, de sorte que les titres complets des signaux/images/groupes soient plus facilement lisibles
- Fonctionnalité de suppression de métadonnées :
  - Avant cette version, la fonctionnalité supprimait toutes les métadonnées, y compris les métadonnées des régions d'intérêt (ROI), le cas échéant.
  - A présent, une boîte de dialogue de confirmation est affichée à l'utilisateur avant de supprimer toutes les métadonnées si le signal/l'image a des métadonnées ROI : cela permet de conserver les métadonnées ROI si nécessaire.
- Extraction de profil d'image : ajout de la prise en charge des images masquées (lors de la définition des régions d'intérêt, les zones en dehors des ROI sont masquées, et le profil est extrait uniquement sur les zones non masquées, ou moyenné sur les zones non masquées dans le cas de l'extraction du profil moyen)
- Style de courbe : ajout de « Réinitialiser les styles de courbe » dans le menu « Affichage ». Cette fonctionnalité permet de réinitialiser le cycle de style de courbe à son état initial.
- Classe de base des plugins *PluginBase* :
  - Ajout de la méthode *edit\_new\_signal\_parameters* pour afficher une boîte de dialogue pour éditer les paramètres d'un nouveau signal
  - Ajout de la méthode *edit\_new\_image\_parameters* pour afficher une boîte de dialogue pour éditer les paramètres d'une nouvelle image (mise à jour du plugin *cdl\_testdata.py* en conséquence)
- API de calculs sur les signaux et les images (*cdl.computations*) :
  - Ajout de wrappers pour les calculs 1 -> 1 sur les signaux et les images
  - Ces wrappers visent à simplifier la création d'une fonction de calcul de base opérant sur les objets natifs de DataLab (*SignalObj* et *ImageObj*) à partir d'une fonction opérant sur des tableaux NumPy
  - Cela simplifie les fonctionnalités internes de DataLab et facilite la création de nouvelles fonctionnalités de calcul dans les plugins
  - Voir le plugin d'exemple *cdl\_custom\_func.py* pour un cas d'utilisation pratique
- Ajout de la fonctionnalité « Extraction du profil radial » au menu « Opération » du panneau d'image :
  - Cette fonctionnalité permet d'extraire un profil moyenné radialement à partir d'une image
  - Le profil est extrait autour d'un centre défini par l'utilisateur (x0, y0)
  - Le centre peut également être calculé (centre de gravité ou centre de l'image)
- Suite de tests automatisés :
  - Depuis la version 0.10, l'objet proxy de DataLab dispose d'une méthode *toggle\_auto\_refresh* pour activer/désactiver la fonctionnalité « Auto-refresh ». Cette fonctionnalité peut être utile pour améliorer les performances lors de l'exécution de scripts de test
  - Les scénarios de test sur les signaux et les images utilisent désormais cette fonctionnalité pour améliorer les performances
- Métadonnées des signaux et des images :
  - Ajout de l'entrée « source » dans le dictionnaire de métadonnées, pour stocker le chemin du fichier source lors de l'importation d'un signal ou d'une image à partir d'un fichier
  - Ce champ est conservé lors du traitement du signal/image, afin de garder une trace du chemin du fichier source

#### Documentation :

- Nouvelle [section Tutoriels](<https://datalab-platform.com/fr/intro/tutorials/index.html>) dans la documentation :
  - Cette section fournit un ensemble de tutoriels pour apprendre à utiliser DataLab
  - Les tutoriels vidéo suivants sont disponibles :
    - Démo rapide
    - Ajout de vos propres fonctionnalités
  - Les tutoriels textuels suivants sont disponibles :
    - Traitement d'un spectre

- Détection de taches sur une image
- Mesure de franges Fabry-Perot
- Prototypage d'un pipeline de traitement personnalisé
- Nouvelle [section API](<https://datalab-platform.com/fr/api/index.html>) dans la documentation :
  - Cette section explique comment utiliser DataLab comme une bibliothèque Python, en abordant les sujets suivants :
    - Comment utiliser les algorithmes DataLab sur les tableaux NumPy
    - Comment utiliser les fonctionnalités de calcul de DataLab sur les objets DataLab (signaux et images)
    - Comment utiliser les fonctionnalités d'entrée/sortie de DataLab
    - Comment utiliser les objets proxy pour contrôler DataLab à distance
  - Cette section fournit également une référence complète de l'API pour les objets et les fonctionnalités de DataLab
  - Ceci corrige l'[Issue #19](<https://github.com/DataLab-Platform/DataLab/issues/19>) - Ajout de la documentation de l'API (modèle de données, fonctions sur les tableaux ou les objets de signaux/images, ...)

Corrections de bugs :

- Ceci corrige l'[Issue #29](<https://github.com/DataLab-Platform/DataLab/issues/29>) - Erreur d'ajustement polynomial : *QDialog [...] argument 1 has an unexpected type "SignalProcessor"*
- Fonctionnalité d'extraction de ROI d'image :
  - Avant cette version, lors de l'extraction d'une seule ROI circulaire d'une image avec l'option « Extraire toutes les ROI dans un seul objet d'image » activée, le résultat était une seule image sans le masque de ROI (le masque de ROI n'était disponible que lors de l'extraction de ROI avec l'option désactivée)
  - Cela conduisait à un comportement inattendu, car on pouvait interpréter le résultat (une image carrée sans le masque de ROI) comme le résultat d'une seule ROI rectangulaire
  - A présent, lors de l'extraction d'une seule ROI circulaire d'une image avec l'option « Extraire toutes les ROI dans un seul objet d'image » activée, le résultat est une seule image avec le masque de ROI (comme si l'option était désactivée)
  - Ceci corrige l'[Issue #31](<https://github.com/DataLab-Platform/DataLab/issues/31>) - Extraction d'une seule ROI circulaire : bascule automatique vers la fonction `extract_single_roi`
- Analyse sur ROI circulaire :
  - Avant cette version, lors de l'exécution de calculs sur une ROI circulaire, les résultats étaient inattendus en termes de coordonnées (les résultats semblaient être calculés dans une région située au-dessus de la ROI réelle).
  - Cela était dû à une régression introduite dans une version antérieure.
  - A présent, lors de la définition d'une ROI circulaire et de l'exécution de calculs sur celle-ci, les résultats sont calculés sur la ROI réelle
  - Ceci corrige l'[Issue #33](<https://github.com/DataLab-Platform/DataLab/issues/33>) - Analyse sur ROI circulaire : résultats inattendus
- Détection des contours sur ROI :
  - Avant cette version, lors de l'exécution de la détection des contours sur une ROI, certains contours étaient détectés en dehors de la ROI (cela peut être dû à une limitation de la fonction `find_contours` de scikit-image).
  - A présent, grâce à une solution de contournement, les contours erronés sont filtrés.
  - Un nouveau module de test `cdl.tests.features.images.contour_fabryperot_app` a été ajouté pour tester la fonctionnalité de détection des contours sur une image Fabry-Perot (merci à [[@emarin2642](https://github.com/emarin2642)](<https://github.com/emarin2642>) pour sa contribution)
  - Ceci corrige l'[Issue #34](<https://github.com/DataLab-Platform/DataLab/issues/34>) - Détection des contours : résultats inattendus en dehors de la ROI
- Fusion des résultats d'analyse :
  - Avant cette version, lors d'un calcul  $I \rightarrow N$  (somme, moyenne, produit) sur un groupe de signaux/images, les résultats d'analyse associés à chaque signal/image étaient fusionnés en un seul résultat, mais seul le type de résultat présent dans le premier signal/image était conservé.
  - A présent, les résultats d'analyse associés à chaque signal/image sont fusionnés en un seul résultat, quel que soit le type de résultat.
- Ceci corrige l'[Issue #36](<https://github.com/DataLab-Platform/DataLab/issues/36>) - L'état d'activation de l'action « Tout supprimer » n'est parfois pas rafraîchi

- Inversion X/Y de l'image : lors de l'inversion des axes X et Y, les régions d'intérêt (ROI) n'étaient ni supprimées ni inversées (les ROI sont désormais supprimées, jusqu'à ce que nous implémentions la fonction d'inversion, si demandé)
- Groupe « Propriétés » : le bouton « Appliquer » était activé par défaut, même lorsqu'aucune propriété n'avait été modifiée, ce qui était déroutant pour l'utilisateur (le bouton « Appliquer » est désormais désactivé par défaut, et n'est activé que lorsqu'une propriété est modifiée)
- Correction de la méthode *get\_object* du proxy lorsqu'il n'y a pas d'objet à retourner (*None* est retourné au lieu d'une exception)
- Correction de *IndexError : list index out of range* lors de certaines opérations ou calculs sur des groupes de signaux/images (par exemple « Extraction de ROI », « Détection de pics », « Redimensionnement », etc.)
- Glisser-déposer depuis un gestionnaire de fichiers : les noms de fichiers sont désormais triés par ordre alphabétique

### DataLab Version 0.10.1

*Note* : la version 0.10.0 a été presque immédiatement remplacée par la version 0.10.1 en raison d'une correction de bogue de dernière minute

Nouvelles fonctionnalités :

- Fonctionnalités communes aux signaux et aux images :
  - Ajout des fonctionnalités « Partie réelle » et « Partie imaginaire » au menu « Opération »
  - Ajout de la fonctionnalité « Convertir le type de données » au menu « Opération »
- Fonctionnalités ajoutées suite aux demandes des utilisateurs (rencontre du 18/12/2023 au CEA) :
  - Les styles de courbe et d'image sont désormais enregistrés dans le fichier HDF5 :
    - Le style de courbe couvre les propriétés suivantes : couleur, style de ligne, largeur de ligne, style de marqueur, taille de marqueur, couleur de bordure de marqueur, couleur de remplissage de marqueur, etc.
    - Le style d'image couvre les propriétés suivantes : colormap, interpolation, etc.
    - Ces propriétés étaient déjà persistantes pendant la session de travail, mais étaient perdues lors de l'enregistrement et du rechargement du fichier HDF5
    - Désormais, ces propriétés sont enregistrées dans le fichier HDF5 et sont restaurées lors du rechargement du fichier HDF5
  - Nouvelles fonctionnalités d'extraction de profil pour les images :
    - Ajout de la fonctionnalité « Profil rectiligne » au menu « Opérations », pour extraire un profil d'une image le long d'une ligne ou d'une colonne
    - Ajout de la fonctionnalité « Profil moyen » au menu « Opérations », pour extraire le profil moyen sur une zone rectangulaire d'une image, le long d'une ligne ou d'une colonne
  - La plage LUT de l'image (paramètres de contraste/luminosité) est désormais enregistrée dans le fichier HDF5 :
    - Comme pour les styles de courbe et d'image, la plage LUT était déjà persistante pendant la session de travail, mais était perdue lors de l'enregistrement et du rechargement du fichier HDF5
    - Désormais, la plage LUT est enregistrée dans le fichier HDF5 et est restaurée lors du rechargement
  - Ajout des actions « Rafraîchissement automatique » et « Rafraîchissement manuel » dans le menu « Affichage » (et la barre d'outils principale) :
    - Lorsque « Rafraîchissement automatique » est activé (par défaut), la vue du graphique est automatiquement rafraîchie lorsqu'un signal/image est modifié, ajouté ou supprimé. Même si le rafraîchissement est optimisé, cela peut entraîner des problèmes de performances lors de l'utilisation de grands ensembles de données.
    - Lorsqu'il est désactivé, la vue du graphique n'est pas automatiquement rafraîchie. L'utilisateur doit rafraîchir manuellement la vue du graphique en cliquant sur le bouton « Rafraîchir manuellement » de la barre d'outils principale ou en appuyant sur la touche de rafraîchissement standard (par exemple « F5 »).
  - Ajout de la méthode *toggle\_auto\_refresh* à l'objet proxy de DataLab :

- Cette méthode permet d'activer/désactiver la fonctionnalité « Rafraîchissement automatique » à partir d'une macro-commande, d'un plugin ou d'un client de contrôle à distance.
- Un gestionnaire de contexte `context_no_refresh` est également disponible pour désactiver temporairement la fonctionnalité « Rafraîchissement automatique » à partir d'une macro-commande, d'un plugin ou d'un client de contrôle à distance. Utilisation typique :

```
with proxy.context_no_refresh():
    # Do something without refreshing the plot view
    proxy.compute_fft() # (...)    ``
```

- Amélioration de la lisibilité des courbes :
    - Jusqu'à cette version, le style de la courbe était automatiquement défini en faisant défiler les styles prédéfinis de **PlotPy**
    - Cependant, certains styles ne sont pas adaptés à la lisibilité des courbes (par exemple, les couleurs « cyan » et « jaune » ne sont pas lisibles sur un fond blanc, en particulier lorsqu'elles sont combinées avec un style de ligne « pointillée »)
    - Cette version introduit une nouvelle gestion du style de courbe avec des couleurs qui sont distinguables et accessibles, même pour les personnes atteintes de déficience de la vision des couleurs
  - Ajout de la fonctionnalité « Anti-aliasing de la courbe » au menu « Affichage » (et à la barre d'outils) :
    - Cette fonctionnalité permet d'activer/désactiver l'anti-aliasing de la courbe (par défaut : activé)
    - Lorsqu'il est activé, le rendu de la courbe est plus lisse mais peut entraîner des problèmes de performances lors de l'utilisation de grands ensembles de données (c'est pourquoi il peut être désactivé)
  - Ajout de la méthode `toggle_show_titles` à l'objet proxy de DataLab. Cette méthode permet d'activer/désactiver la fonctionnalité « Afficher les titres des objets graphiques » à partir d'une macro-commande, d'un plugin ou d'un client de contrôle à distance.
  - Le client distant vérifie désormais la version du serveur et affiche un message d'avertissement si la version du serveur n'est peut-être pas entièrement compatible avec la version du client.
- Corrections de bugs :
- Fonctionnalité de détection des contours de l'image (menu Analyser ») :
    - La fonctionnalité de détection des contours ne prenait pas en compte le paramètre « shape » (cercle, ellipse, polygone) lors du calcul des contours. Le paramètre était stocké mais vraiment utilisé uniquement lors de l'appel de la fonctionnalité une deuxième fois.
    - Ce comportement involontaire a conduit à une `AssertionError` lors du choix de « polygone » comme forme de contour et de la tentative de calcul des contours pour la première fois.
    - Ceci est maintenant corrigé (voir [Issue #9](https://github.com/DataLab-Platform/DataLab/issues/9)) - Détection des contours de l'image : `AssertionError` lors du choix de « polygone » comme forme de contour)
  - Raccourcis clavier :
    - Les raccourcis clavier pour les actions « Nouveau », « Ouvrir », « Enregistrer », « Dupliquer », « Supprimer », « Tout supprimer » et « Rafraîchir manuellement » ne fonctionnaient pas correctement.
    - Ces raccourcis étaient spécifiques à chaque panneau de signal/image et ne fonctionnaient que lorsque le panneau sur lequel le raccourci a été pressé pour la première fois était actif (lorsqu'il était activé à partir d'un autre panneau, le raccourci ne fonctionnait pas et un message d'avertissement était affiché dans la console, par exemple `QAction : :event : Ambiguous shortcut overload : Ctrl+C`)
    - De plus, les raccourcis ne fonctionnaient pas au démarrage (lorsqu'aucun panneau n'avait le focus).
    - Ceci est maintenant corrigé : les raccourcis fonctionnent maintenant quel que soit le panneau actif, et même au démarrage (voir [Issue #10](https://github.com/DataLab-Platform/DataLab/issues/10)) - Raccourcis clavier ne fonctionnant pas correctement : `QAction : :event : Ambiguous shortcut overload : Ctrl+C`)
  - Les actions « Afficher les titres des objets graphiques » et « Rafraîchissement automatique » ne fonctionnaient pas correctement :
    - Les actions « Afficher les titres des objets graphiques » et « Rafraîchissement automatique » ne fonctionnaient que sur le panneau de signal/image actif, et non sur tous les panneaux.
    - Ceci est maintenant corrigé (voir [Issue #11](https://github.com/DataLab-Platform/DataLab/issues/11)) - Les actions « Afficher les titres des objets graphiques » et « Rafraîchissement automatique » ne fonctionnaient que sur le panneau de signal/image actuel)

- Correction de l'[Issue #14](https://github.com/DataLab-Platform/DataLab/issues/14) - Enregistrer/Réouvrir le projet HDF5 sans nettoyage entraîne une *ValueError*
- Correction de l'[Issue #15](https://github.com/DataLab-Platform/DataLab/issues/15) - MacOS : 1. Erreur *pip install cdl* - 2. Menus manquants :
  - Partie 1 : l'erreur *pip install cdl* sur MacOS était en fait un problème de **PlotPy** (voir [ce problème])
  - Partie 2 : les menus manquants sur MacOS étaient dus à un bogue PyQt/MacOS concernant les menus dynamiques
- Format de fichier HDF5 : lors de l'importation d'un ensemble de données HDF5 en tant que signal ou image, les attributs de l'ensemble de données étaient systématiquement copiés dans les métadonnées du signal/image : nous ne copions désormais que les attributs qui correspondent aux types de données standard (entiers, flottants, chaînes de caractères) pour éviter les erreurs lors de la sérialisation/désérialisation de l'objet signal/image
- Visualiseur d'installation/configuration : amélioration de la lisibilité (suppression de la coloration syntaxique)
- Fichier de spécification de PyInstaller : ajout manuel des fichiers de données *skimage* manquants afin de continuer à prendre en charge Python 3.8 (voir [Issue #12](https://github.com/DataLab-Platform/DataLab/issues/12) - Version autonome sur Windows 7 : *api-ms-win-core-path-l1-1-0.dll* manquant)
- Correction de l'[Issue #13](https://github.com/DataLab-Platform/DataLab/issues/13) - ArchLinux : *qt.qpa.plugin : Could not load the Qt platform plugin « xcb » in « » even though it was found*

## DataLab Version 0.9.2

### Corrections de bugs :

- Fonctionnalité d'extraction de la région d'intérêt (ROI) pour les images :
  - L'extraction de la ROI ne fonctionnait pas correctement lorsque l'option « Extraire toutes les ROI dans un seul objet image » était activée s'il n'y avait qu'une seule ROI définie. Le résultat était une image positionnée à l'origine (0, 0) au lieu de la position attendue (x0, y0) et le rectangle de la ROI lui-même n'était pas supprimé comme prévu. Ceci est maintenant corrigé (voir [Issue #6](https://github.com/DataLab-Platform/DataLab/issues/6) - Fonctionnalité "Extraire plusieurs ROI" : résultat inattendu pour une seule ROI)
  - Les rectangles ROI avec des coordonnées négatives n'étaient pas correctement gérés : l'extraction de la ROI levait une exception *ValueError*, et le masque de l'image n'était pas correctement affiché. Ceci est maintenant corrigé (voir [Issue #7](https://github.com/DataLab-Platform/DataLab/issues/7) - Extraction de la ROI de l'image : *ValueError : zero-size array to reduction operation minimum which has no identity*)
  - L'extraction de la ROI ne prenait pas en compte la taille des pixels (dx, dy) et l'origine (x0, y0) de l'image. Ceci est maintenant corrigé (voir [Issue #8](https://github.com/DataLab-Platform/DataLab/issues/8) - Extraction de la ROI de l'image : prendre en compte la taille des pixels)
- La console de macro-commande est désormais en lecture seule :
  - La console Python du panneau de macro-commande ne prend actuellement pas en charge le flux d'entrée standard (*stdin*) et c'est voulu (du moins pour l'instant)
  - Définir la console Python en lecture seule pour éviter toute confusion

## DataLab Version 0.9.1

### Corrections de bugs :

- La traduction française n'est pas disponible sur Windows/Version autonome :
  - La locale n'était pas correctement détectée sur Windows pour la version autonome (gelée avec *pyinstaller*) en raison d'un problème avec *locale.getlocale()* (fonction renvoyant *None* au lieu de la locale attendue sur les applications gelées)
  - C'est finalement un problème de *pyinstaller*, mais une solution de contournement a été implémentée dans *guidata* V3.2.2 (voir [Issue guidata #68](https://github.com/PlotPyStack/guidata/issues/68) - Windows : la traduction gettext ne fonctionne pas sur les applications gelées)
  - [Issue #2](https://github.com/DataLab-Platform/DataLab/issues/2) - La traduction française n'est pas disponible sur la version autonome de Windows
- L'enregistrement d'une image au format JPEG2000 échoue pour les données non entières :
  - L'encodeur JPEG2000 ne prend pas en charge les données non entières ou les données entières signées

- Avant, DataLab affichait un message d'erreur lors de la tentative de sauvegarde de données incompatibles au format JPEG2000 : cela n'était pas cohérent avec le comportement des autres formats d'image standard (par exemple PNG, JPG, etc.) pour lesquels DataLab convertissait automatiquement les données au format approprié (entier non signé sur 8 bits)
- Le comportement actuel est maintenant cohérent avec les autres formats d'image standard : lors de la sauvegarde au format JPEG2000, DataLab convertit automatiquement les données en entier non signé sur 8 bits ou en entier non signé sur 16 bits (en fonction du type de données original)
- [Issue #3](<https://github.com/DataLab-Platform/DataLab/issues/3>) - Sauvegarde d'image au format JPEG2000 : "OSError : erreur d'encodeur -2 lors de l'écriture du fichier image"
- Les raccourcis de la version autonome de Windows ne s'affichent pas dans le menu de démarrage de l'utilisateur actuel :
  - Lors de l'installation de DataLab sur Windows à partir d'un compte non administrateur, les raccourcis ne s'affichaient pas dans le menu de démarrage de l'utilisateur actuel, mais dans le menu de démarrage de l'administrateur (en raison des privilèges élevés de l'installateur et du fait que l'installateur ne prend pas en charge l'installation de raccourcis pour tous les utilisateurs)
  - Désormais, l'installateur ne demande plus de privilèges élevés, et les raccourcis sont installés dans le menu de démarrage de l'utilisateur actuel (cela signifie également que l'utilisateur actuel doit disposer d'un accès en écriture au répertoire d'installation)
  - Dans les futures versions, l'installateur prendra en charge l'installation de raccourcis pour tous les utilisateurs s'il y a une demande à cet effet (voir [Issue #5](<https://github.com/DataLab-Platform/DataLab/issues/5>))
  - [Issue #4](<https://github.com/DataLab-Platform/DataLab/issues/4>) - Windows : les raccourcis de la version autonome ne s'affichent pas dans le menu de démarrage de l'utilisateur actuel
- Fenêtre d'installation et de configuration pour la version autonome :
  - Ne plus afficher le message d'erreur ambigu "Dépendances invalides"
  - Les dépendances sont censées être vérifiées lors de la construction de la version autonome
- Ajout de la documentation PDF à la version autonome :
  - La documentation PDF était manquante dans la version précédente
  - Désormais, la documentation PDF (en anglais et en français) est incluse dans la version autonome

## DataLab Version 0.9.0

### Nouvelles dépendances :

- DataLab est désormais alimenté par [PlotPyStack](<https://github.com/PlotPyStack>) :
  - PythonQwt
  - guidata
  - PlotPy
- [opencv-python](<https://pypi.org/project/opencv-python/>) (algorithmes de traitement d'image)

### Nouvelle plateforme de référence :

- DataLab est validé sur Windows 11 avec Python 3.11 et PyQt 5.15
- DataLab est également compatible avec d'autres systèmes d'exploitation (Linux, MacOS) et d'autres liaisons Python-Qt et versions (Python 3.8-3.12, PyQt6, PySide6)

### Nouvelles fonctionnalités :

- DataLab est une plateforme :
  - Ajout de la prise en charge des plugins
    - Fonctionnalités de traitement personnalisées disponibles dans le menu « Plugins »
    - Fonctionnalités d'E/S personnalisées : de nouveaux formats de fichier peuvent être ajoutés aux fonctionnalités d'E/S standard pour les signaux et les images
    - Fonctionnalités HDF5 personnalisées : de nouveaux formats de fichier HDF5 peuvent être ajoutés à la fonctionnalité d'importation HDF5 standard
    - D'autres fonctionnalités à venir...
- Ajout de la fonctionnalité de contrôle à distance : DataLab peut être contrôlé à distance via une connexion TCP/IP (voir [Contrôle à distance]([https://datalab-platform.com/fr/remote\\_control.html](https://datalab-platform.com/fr/remote_control.html)))

- Ajout de commandes de macro : DataLab peut être contrôlé via un fichier macro (voir [Commandes de macro](https://datalab-platform.com/fr/macro\_commands.html))
- Fonctionnalités générales :
  - Ajout de la boîte de dialogue des paramètres (voir l'entrée « Paramètres » dans le menu « Fichier ») :
    - Paramètres généraux
    - Paramètres de visualisation
    - Paramètres de traitement
    - Etc.
  - Nouvelle disposition par défaut : les panneaux de signaux/images sont sur le côté droit de la fenêtre principale, les panneaux de visualisation sont sur le côté gauche avec une barre d'outils verticale
- Fonctionnalités de signal/image :
  - Ajout de l'isolation des processus : chaque signal/image est traité dans un processus séparé, de sorte que DataLab ne se fige plus lors du traitement de signaux/images volumineux
  - Ajout de la prise en charge des groupes : les signaux et les images peuvent être regroupés, et des opérations peuvent être appliquées à tous les objets d'un groupe ou entre les groupes
  - Ajout de boîtes de dialogue d'avertissement et d'erreur avec des liens de poursuite détaillés vers le code source (les avertissements peuvent être ignorés en option)
  - Amélioration significative des performances lors de la sélection d'objets
  - Optimisation des performances lors de l'affichage d'images volumineuses
  - Ajout de la prise en charge du dépôt de fichiers sur le panneau de signal/image
  - Ajout de la boîte de groupe « Paramètres de calcul » pour afficher les paramètres d'entrée du dernier résultat
  - Ajout de la fonctionnalité « Copier les titres dans le presse-papiers » dans le menu « Édition »
  - Pour chaque fonctionnalité de traitement individuelle (menus opération, traitement et analyse), les paramètres saisis (boîtes de dialogue) sont stockés en cache pour être utilisés par défaut la prochaine fois que la fonctionnalité est utilisée
- Traitement de signal :
  - Ajout de la prise en charge du décalage FFT facultatif (voir la boîte de dialogue des paramètres)
- Traitement d'image :
  - Ajout de l'opération de binning de pixels (facteurs de binning X/Y, opération : somme, moyenne, ...)
  - Ajout des options « Distribuer sur une grille » et « Réinitialiser les positions des images » dans le menu des opérations
  - Ajout du filtre de Butterworth
  - Ajout des fonctionnalités de traitement de l'exposition :
    - Correction gamma
    - Correction logarithmique
    - Correction sigmoïde
  - Ajout des fonctionnalités de traitement de la restauration :
    - Filtre de débruitage par variation totale (TV Chambolle)
    - Filtre bilatéral (débruitage)
    - Filtre de débruitage par ondelettes
    - Filtre de débruitage White Top-Hat
  - Ajout des transformations morphologiques (empreinte de disque) :
    - White Top-Hat
    - Black Top-Hat
    - Érosion
    - Dilatation
    - Ouverture
    - Fermeture
  - Ajout des fonctionnalités de détection des contours :
    - Filtre de Roberts
    - Filtre de Prewitt (vertical, horizontal, les deux)
    - Filtre de Sobel (vertical, horizontal, les deux)
    - Filtre de Scharr (vertical, horizontal, les deux)
    - Filtre de Farid (vertical, horizontal, les deux)

- Filtre de Laplace
- Filtre de Canny
- Détection des contours : ajout de la prise en charge des contours polygonaux (en plus des contours de cercle et d'ellipse)
- Ajout de la transformation de Hough pour les cercles (détection de cercles)
- Ajout du rééchantillonnage des niveaux d'intensité de l'image
- Ajout de l'égalisation d'histogramme
- Ajout de l'égalisation d'histogramme adaptative
- Ajout des méthodes de détection de blobs :
  - Différence de Gaussienne
  - Méthode du déterminant de Hessian
  - Laplacien de Gaussienne
  - Détection de blobs à l'aide d'OpenCV
- Les formes et les annotations des résultats sont maintenant transformées (au lieu d'être supprimées) lors de l'exécution de l'une des opérations suivantes :
  - Rotation (angle arbitraire, +90°, -90°)
  - Symétrie (verticale/horizontale)
- Ajout de la prise en charge du décalage FFT facultatif (voir la boîte de dialogue des paramètres)
- Console : ajout d'un éditeur externe configurable (par défaut : VSCode) pour suivre les liens de poursuite vers le code source

---

**Note :** DataLab a été créé par [Codra/Pierre Raybaut](#) en 2023. Il est développé et maintenu par l'équipe de développement de la plateforme DataLab.

---

### C

- `cdl.algorithms`, 195
- `cdl.algorithms.coordinates`, 208
- `cdl.algorithms.datatypes`, 207
- `cdl.algorithms.image`, 202
- `cdl.algorithms.signal`, 196
- `cdl.computation`, 282
- `cdl.computation.base`, 283
- `cdl.computation.image`, 308
- `cdl.computation.image.detection`, 340
- `cdl.computation.image.edges`, 337
- `cdl.computation.image.exposure`, 328
- `cdl.computation.image.morphology`, 336
- `cdl.computation.image.restoration`, 332
- `cdl.computation.image.threshold`, 326
- `cdl.computation.signal`, 288
- `cdl.core.gui`, 366
- `cdl.core.gui.actionhandler`, 389
- `cdl.core.gui.docks`, 415
- `cdl.core.gui.h5io`, 416
- `cdl.core.gui.main`, 367
- `cdl.core.gui.objectview`, 393
- `cdl.core.gui.panel`, 373
- `cdl.core.gui.panel.base`, 373
- `cdl.core.gui.panel.image`, 382
- `cdl.core.gui.panel.macro`, 386
- `cdl.core.gui.panel.signal`, 379
- `cdl.core.gui.plothandler`, 396
- `cdl.core.gui.processor`, 400
- `cdl.core.gui.processor.base`, 400
- `cdl.core.gui.processor.image`, 408
- `cdl.core.gui.processor.signal`, 405
- `cdl.core.gui.roieditor`, 399
- `cdl.obj`, 248
- `cdl.param`, 209
- `cdl.plugins`, 125
- `cdl.proxy`, 349