Data analysis for EELS and CL using HyperSpy

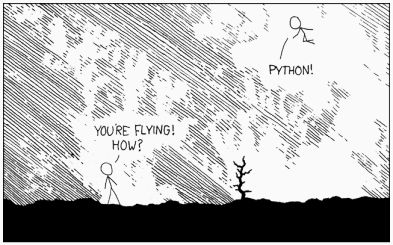
Jonas Lähnemann, Sean Collins, Kagiso Loeto, Domenik Spallek, Aidan Campbell,
... and the HyperSpy, LumiSpy and eXSpy developers

eBEAM 2024 – School on Nano-optics with Free Electrons, Aussois, September 04, 2024

Today: Introductory lecture – complemented by interactive tutorials in small groups

Why use python for scientific data analysis?

- Both a genuine programming and a scripting language
- Free as in speech (open source) and as in beer (no fees)
- Portable: Runs on any environment (Win, Mac, Linux)
- Intuitive and readable code
- Reproducible and reusable analysis routines

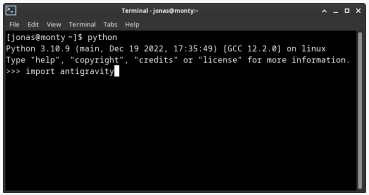


Courtesy of XKCD: https://xkcd.com/353/

python™

Why use python for scientific data analysis?

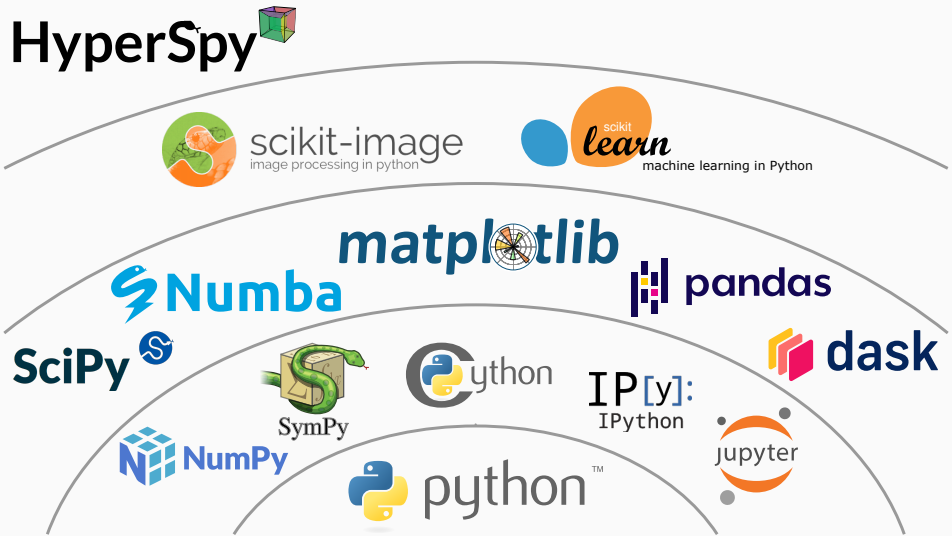
- Both a genuine programming and a scripting language
- Free as in speech (open source) and as in beer (no fees)
- Portable: Runs on any environment (Win, Mac, Linux)
- Intuitive and readable code
- Reproducible and reusable analysis routines
- Different environments
  - Use interactive console: `python`, `ipython`
  - Run script files: `python silly-walk.py`
  - Matlab-style integrated desktop environments: e.g. `spyder`
  - Mathematica-style notebooks: `Jupyter`
  - Dedicated frontends: e.g. `hyperspyUI`



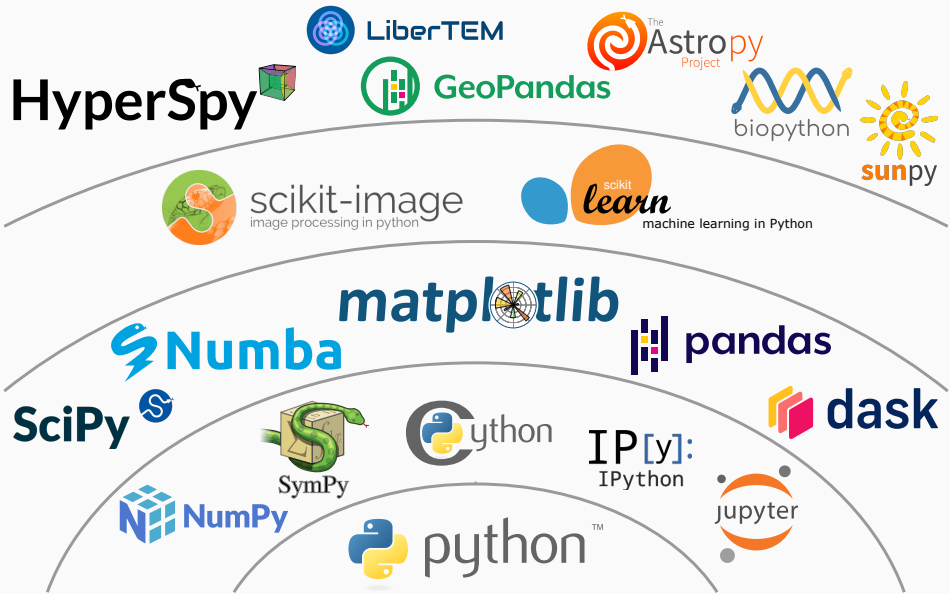
Why use python for scientific data analysis?

- Both a genuine programming and a scripting language
- Free as in speech (open source) and as in beer (no fees)
- Portable: Runs on any environment (Win, Mac, Linux)
- Intuitive and readable code
- Reproducible and reusable analysis routines
- Different environments
- Vast and unmatched range of scientific libraries:
  - `numpy` + `scipy` + `matplotlib` = Matlab replacement
  - `SymPy`, `scikit-image`, `scikit-learn`, ...

The scientific python stack

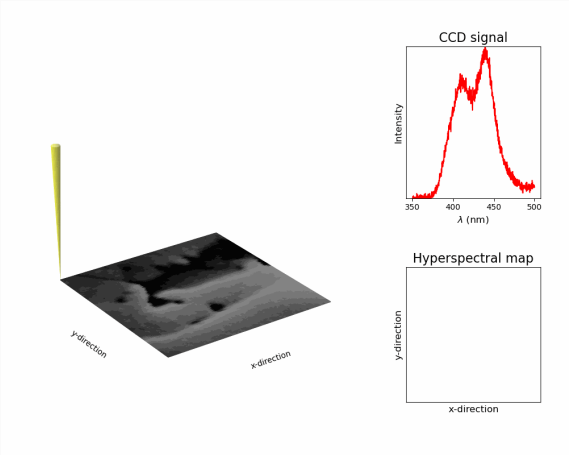


The scientific python stack



What is HyperSpy?

- Open-source `python` library for interactive **analysis of multi-dimensional data**
- Easily extend operations from a single spectrum/image to multi-dimensional arrays



HyperSpy

What is HyperSpy?

- Open-source `python` library for interactive **analysis of multi-dimensional data**
- Easily extend operations from a single spectrum/image to multi-dimensional arrays
- Simple, intuitive syntax
- Easy access to cutting-edge signal processing tools

```
s = hs.load('beautiful-plumage.hspy')
m = s.create_model()
g = hs.model.components1D.GaussianHF()
m.append(g)
m.multifit()
```

HyperSpy

What is HyperSpy?

- Open-source `python` library for interactive **analysis of multi-dimensional data**
- Easily extend operations from a single spectrum/image to multi-dimensional arrays
- Simple, intuitive syntax
- Easy access to cutting-edge signal processing tools
- Community-driven: Developed by scientists for scientists
- Vibrant and welcoming community of developers and users committed to well-documented, powerful and easy-to-use toolbox

Francisco de la Peña 😇
Pierre Burdet
Magnus Nord
Mike Sarahan
Vadim Migunov
Timothy Poon
Suhas Somnath

Eric Prestat
Petras Jokubauskas
Tomas Ostasevicius
Joshua Taillon 👋
Alberto Eljarrat
Stefano Mazzucco
Tom Slater

Vidar Tonaas Fauske
Tom Furnival
Katherine MacArthur
Thomas Aarholt
Jan Caron
Nicolas Tappy
Michael Walls

Jonas Lähnemann
Carter Francis
Duncan N. Johnstone
Paul Quinn
Takashi Nemoto
Niels Cautaerts
Hugh Ramsden

HyperSpy

What is HyperSpy?

- Open-source **python** library for interactive **analysis of multi-dimensional data**
- Easily extend operations from a single spectrum/image to multi-dimensional arrays
- Simple, intuitive syntax
- Easy access to cutting-edge signal processing tools
- Community-driven: Developed by scientists for scientists
- Vibrant and welcoming community of developers and users committed to well-documented, powerful and easy-to-use toolbox
- Mature and robust code base
- Ecosystem: Extension framework for different signal types

HyperSpy

A brief history of HyperSpy?

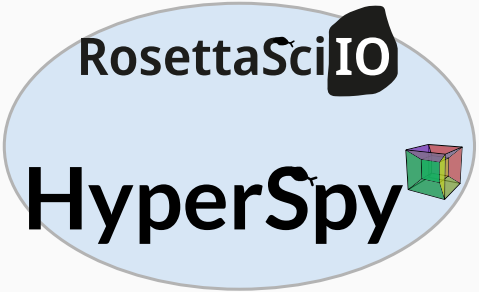
- 2007-2012: Developed by Francisco de la Peña in his PhD (Univ. Paris-Sud) as EELSlab
- 2010: Open-sourced on GitHub and renamed to HyperSpy
- 2016: Migration to python 3 (with version 0.8.4) and release of version 1.0
- 2019: Introduction of extension mechanism (with version 1.5)
- 2023: Version 2.0 released – I/O and domain-specific functions split out

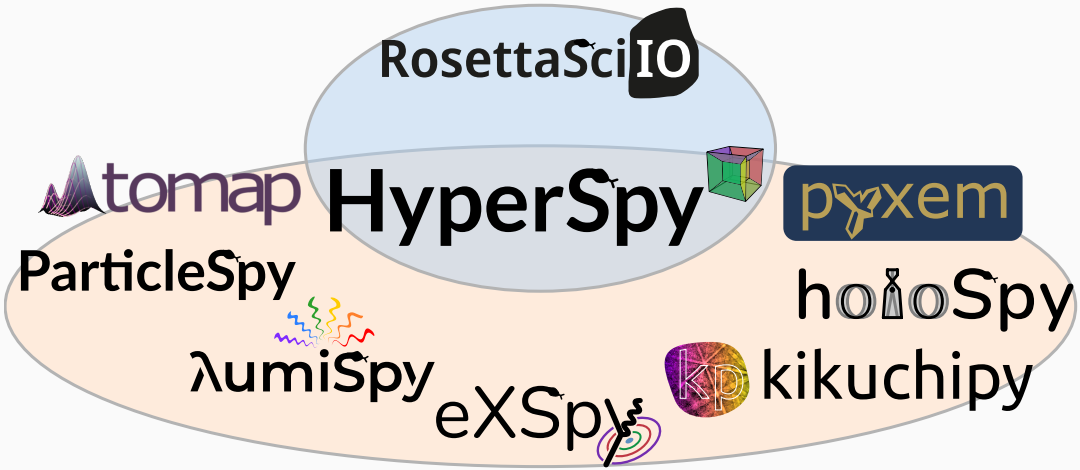
HyperSpy 

A brief history of HyperSpy?

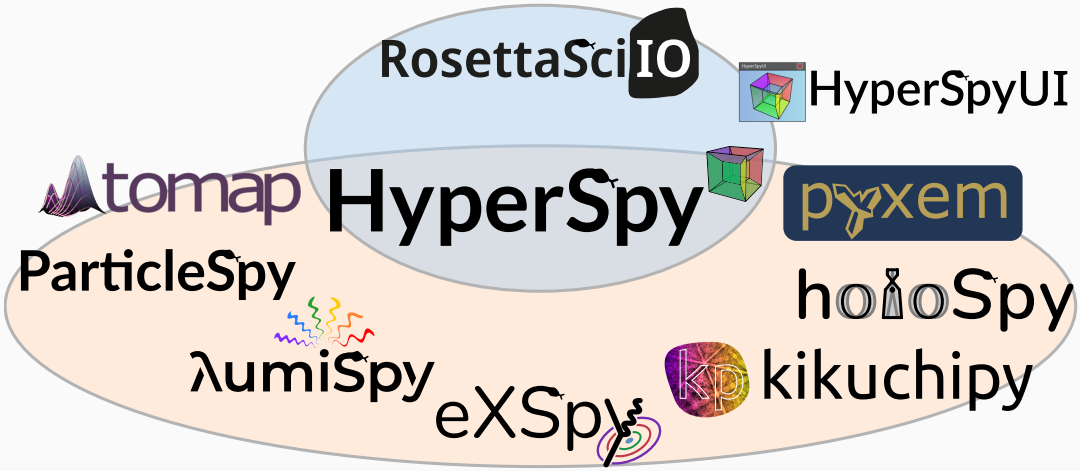
- 2007-2012: Developed by Francisco de la Peña in his PhD (Univ. Paris-Sud) as EELSlab
- 2010: Open-sourced on GitHub and renamed to HyperSpy
- 2016: Migration to python 3 (with version 0.8.4) and release of version 1.0
- 2019: Introduction of extension mechanism (with version 1.5)
- 2023: Version 2.0 released – I/O and domain-specific functions split out
- Now ...
  - $>$ 1200 citations (google scholar)
  - 56 releases
  - 70 contributors
  - $>$ 90k lines of code (excluding extensions)
  - $>$ 500 stars on GitHub (users with GitHub account)
  - used in $>$ 200 other GitHub projects
  - ... and still growing!

HyperSpy 

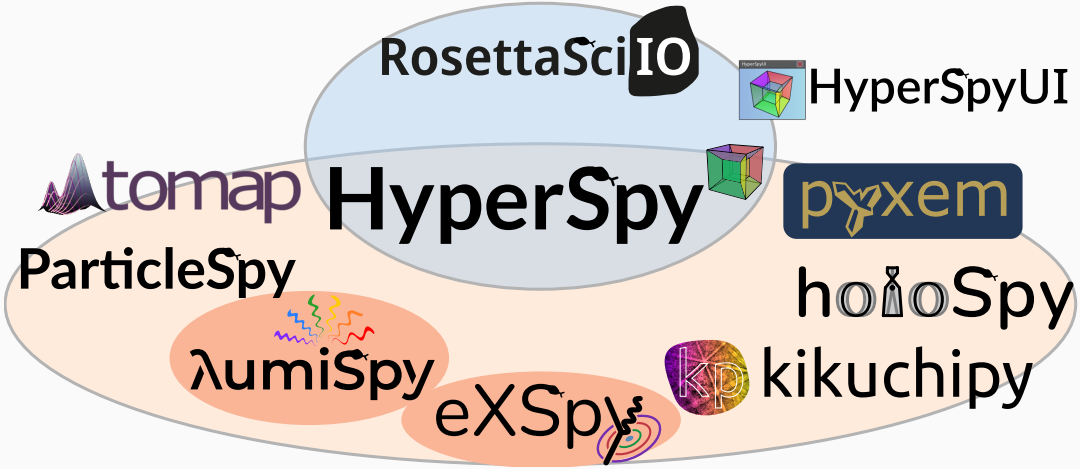

RosettaSciIO
HyperSpy


RosettaSciIO
tomap
HyperSpy
pyxem
ParticleSpy
holoSpy
λumiSpy
eXSpy
kikuchipy

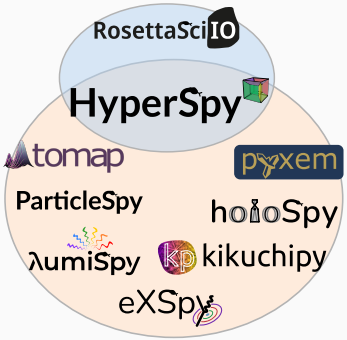
The HyperSpy ecosystem (v2.0)



The HyperSpy ecosystem (v2.0)



The HyperSpy ecosystem: Extensions



- `RosettaSciIO`: Reading and writing scientific data formats

Domain specific:

- `eXSpy`: Energy Dispersive X-ray Spectroscopy (EDS) and Electron Energy Loss Spectroscopy (EELS)
- `LumiSpy`: Luminescence spectroscopy
- `pyxem`: 4D Scanning Transmission Electron Microscopy (STEM)
- `kikuchipy`: Electron Backscatter Diffraction (EBSD) patterns
- `holospy`: Off-axis Electron Holography
- `Atomap`: Atomic resolution STEM
- `ParticleSpy`: Segmentation and analysis of nanoparticles

- `HyperSpyUI`: User Interface (limited functionality)

HyperSpy features include

- Tools for loading/saving various data file formats (`RosettaSciIO`)
- Analytical tools that exploit the multidimensionality of datasets
- Data visualization: evaluate datasets during the analysis, provide interactive operation for certain functions, and plotting
- Efficient handling of big datasets ("lazy" and parallel processing)
- Extracting subsets of data from multidimensional datasets: regions of interest and a powerful numpy-style indexing mechanism
- Handling of non-uniform data axes
- User-friendly and powerful framework for multidimensional model fitting: provides many standard functions, easily extended to custom ones
- Machine learning algorithms, useful for e.g. denoising data, decomposition

HyperSpy 

Selection of supported data formats

For saving analyses, **RosettaSciIO** provides dedicated hdf5-based `.hspy` and compressible zarr `.zspy` data formats.

RosettaSciIO supports a wide range of microscopy (and spectroscopy) related file types.

All relevant formats for cathodoluminescence (CL) spectroscopy can be read:

- Attolight digital surface `.sur` (write support, limited `metadata` parsing)
- Delmic `.hdf5` (full parsing with next release)
- Gatan DigitalMicrograph `.dm3/.dm4`
- Horiba JobinYvon `.xml` (same metadata as binary `.l5s` or `.l6s`)

```
s = hs.load('spam-eggs-tofu.hspy')
```

RosettaSciIO

Selection of supported data formats

For saving analyses, RosettaSciIO provides dedicated hdf5-based `.hspy` and compressible zarr `.zspy` data formats.

RosettaSciIO supports a wide range of microscopy (and spectroscopy) related file types.

Other relevant formats for luminescence spectroscopy:

- Hamamatsu streak camera images in `.img` or `.tif` format
- Renishaw wire format `.wdf`
- TriVista XML `.tvf`

```
s = hs.load('spam-spam-tofu.tif')
```

RosettaSciIO

Selection of supported data formats

For saving analyses, **RosettaSciIO** provides dedicated hdf5-based `.hspy` and compressible zarr `.zspy` data formats.

RosettaSciIO supports a wide range of microscopy (and spectroscopy) related file types.

Relevant formats for EDS and EELS include:

- MSA spectral file format `.msa` (write support)
- Bruker `.spx` (single spectrum) and `.bcf` (spectrum images)
- CEOS Panta Rhei `.prz` (EELS)
- EDAX `.spc` (single spectra) and `.spd` (EDS spectrum image)
- FEI TEM Imaging & Analysis `.ser` and `.emi` files (EELS & EDS spectra/spectrum images)
- Gatan DigitalMicrograph `.dm3/.dm4` (EELS)
- ThermoFisher Velox Electron Microscopy Dataset `.emd` (EDS spectrum images)

```
s = hs.load('spam-eggs-spam.msa')
```

RosettaSciIO

Selection of supported data formats

For saving analyses, RosettaSciIO provides dedicated hdf5-based `.hspy` and compressible zarr `.zspy` data formats.

RosettaSciIO supports a wide range of microscopy (and spectroscopy) related file types.

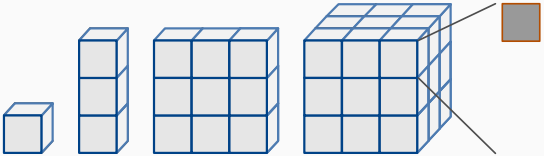
Generic data formats, such as NeXus HDF5 or many image formats are also supported.

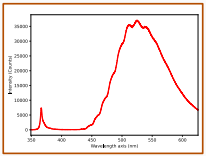
Common 4D-STEM datasets are supported for analysis with `pyxem`.

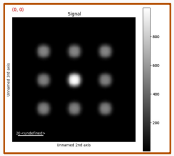
```
s = hs.load('spam-spam-spam.zspy')
```

RosettaSciIO

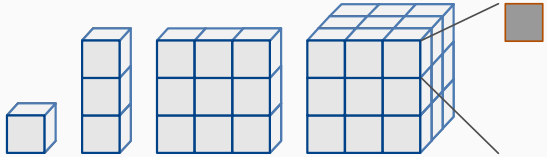
Core design principles: Navigation versus Signal dimensions

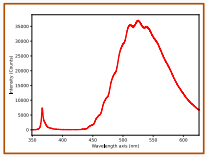


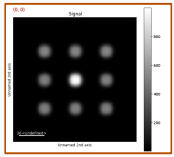


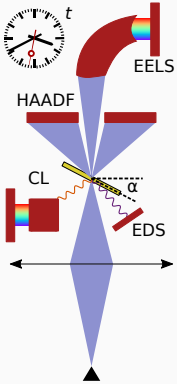


Core design principles: Navigation versus Signal dimensions









[Francisco de la Peña]

Technique	Dimensions Navigation | Signal	Total
HAADF	$(x, y\|)$ / $(\|x, y)$	2
Spectrum (CL, EDS, EELS)	$(\|E)$	1
Linescan (CL, EDS, EELS)	$(x\|E)$	2
Map (CL, EDS, EELS)	$(x, y\|E)$	3
Temperature series map (CL)	$(x, y, T\|E)$	4
Tilt series map (EELS)	$(x, y, \alpha\|E)$	4
Tilt and time series map (EELS)	$(x, y, \alpha, t\|E)$	5
Diffraction	$(x, y, \alpha, t\|x^*, y^*)$	6

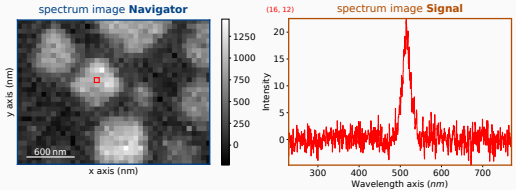
Core design principles: Navigation versus Signal dimensions

- Navigation and signal axes are interchangeable
- Transposing signal flips navigation and signal axes

The same 3D dataset can be represented as either:

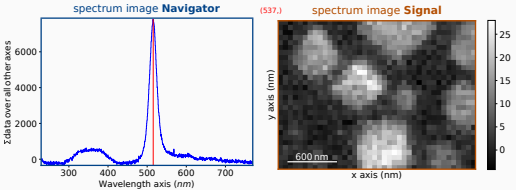
2D array of spectra
$(x, y|\lambda)$

```
cl1.plot()
```

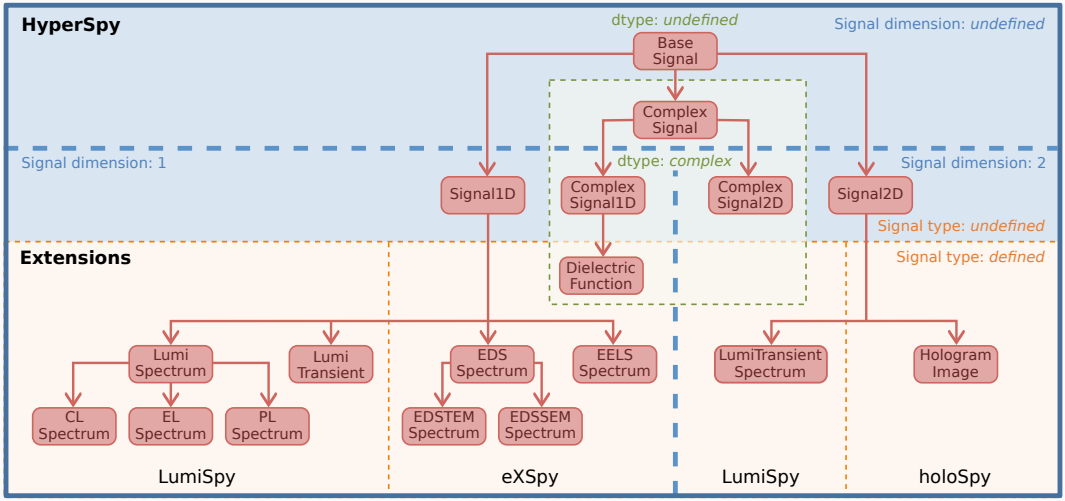


1D array of images
$(\lambda|x, y)$

```
cl1.T.plot()
```

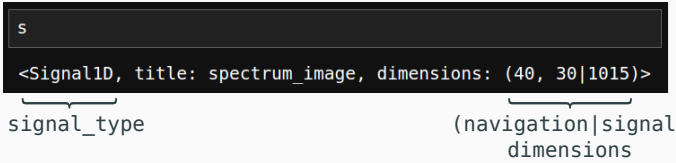


Core design principles: The HyperSpy `signal class` hierarchy



`subclasses` inherit functions & provide additional (datatype/domain-specific) features

Core design principles: HyperSpy signal object



- `Axes manager` – axes objects for each navigation and signal axis
- `Data` – multidimensional array
- `Metadata` – parsed information about measurement conditions, sample and file
- `(Original metadata)` – metadata tree as saved in vendor file

Core design principles: HyperSpy signal object

```
s

<Signal1D, title: spectrum_image, dimensions: (40, 30|1015)>
```

`signal_type` — `(navigation|signal) dimensions`

- `Axes manager` – axes objects for each navigation and signal axis
- `Data` – multidimensional array
- `Metadata` – parsed information about measurement conditions, sample and file
- `(Original metadata)` – metadata tree as saved in vendor file

```
s.axes_manager
```

< Axes manager, axes: (40, 30|1015) >

Navigation axis name	size	index	offset	scale	units
x	40	0	0.0	61.0	nm
y	30	29	0.0	61.0	nm
Signal axis name	**size**		**offset**	**scale**	**units**
Wavelength	1015		229.0	1.0	nm

Core design principles: HyperSpy signal object

```
s

<Signal1D, title: spectrum_image, dimensions: (40, 30|1015)>
```

signal_type

(navigation|signal) dimensions

- `Axes manager` – axes objects for each navigation and signal axis
- `Data` – multidimensional array
- `Metadata` – parsed information about measurement conditions, sample and file
- `(Original metadata)` – metadata tree as saved in vendor file

```
s.data

array([[[-2.49761505e+00, -2.12892204e+00, -6.81961551e-01, ...,
          2.39770962e-01,  3.13369195e+00,  5.80652438e-01],
        [ 2.43140998e+00, -3.19181309e+00, -1.34478831e+00, ...,
         -1.91813092e-01,  1.34926127e+00,  4.33108368e-02],
        [-7.11180500e-01, -1.22476480e+00,  1.16721557e+00, ...,
          2.04561165e+00, -7.11180500e-01, -1.58957657e+00],
        ...,
```

Core design principles: HyperSpy signal object

```
s

<Signal1D, title: spectrum_image, dimensions: (40, 30|1015)>
```

signal_type

(navigation|signal) dimensions

- `Axes manager` – axes objects for each navigation and signal axis
- `Data` – multidimensional array
- `Metadata` – parsed information about measurement conditions, sample and file
- `(Original metadata)` – metadata tree as saved in vendor file

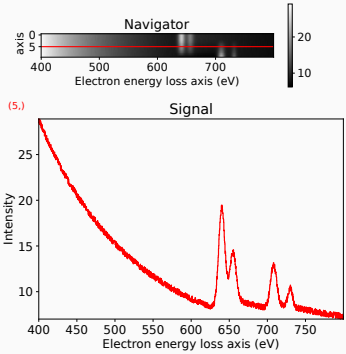
```
s.metadata

▼ Acquisition_instrument
  ▼ CCD
    ◦ amplification = 1
    ◦ binning = 1
    ◦ channels = 1024
    ◦ exposure_time_s = 0.05
  ▼ SEM
▼ General
  ▼ FileIO
    ▶ 0
    ▼ 1
      ◦ hyperspy_version = 2.2.0
      ◦ io_plugin = rsciio.hspy
      ◦ operation = load
```

Core design principles: numpy-style indexing

Simple syntax to select subsets of data in all dimensions:

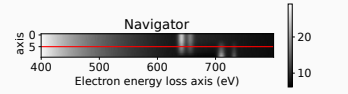
```
s = exspy.data.EELS_MnFe()
s.plot()
```

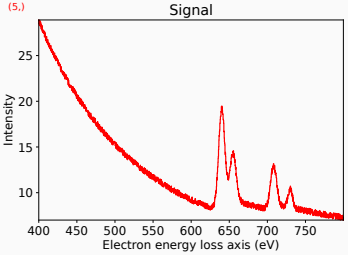


Core design principles: numpy-style indexing

Simple syntax to select subsets of data in all dimensions:

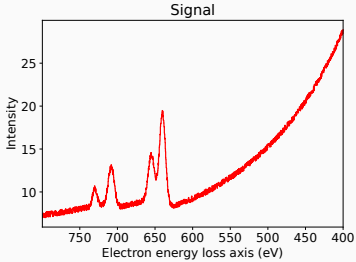
```
s = exspy.data.EELS_MnFe()
s.plot()
```





Select navigation `pixel 5` and invert signal axis:

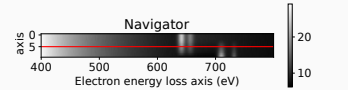
```
s.inav[5].isig[::-1].plot()
```

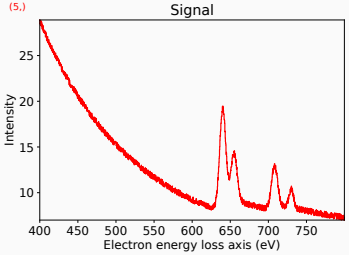


Core design principles: numpy-style indexing

Simple syntax to select subsets of data in all dimensions:

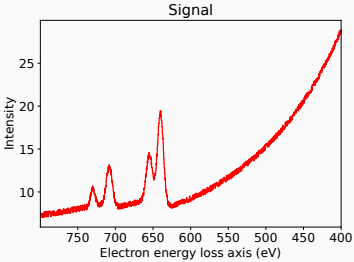
```
s = exspy.data.EELS_MnFe()
s.plot()
```





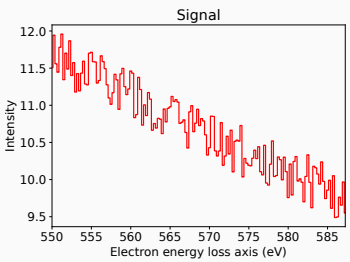
Select navigation `pixel 5` and invert signal axis:

```
s.inav[5].isig[::-1].plot()
```



Plot single spectrum signal range: `pixels 600-750`

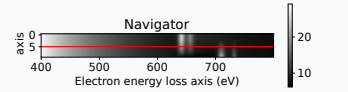
```
s.inav[5].isig[600:750].plot()
```

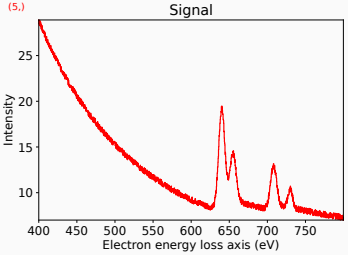


Core design principles: numpy-style indexing

Simple syntax to select subsets of data in all dimensions:

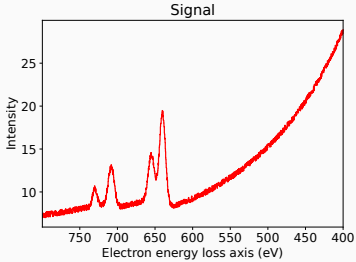
```
s = exspy.data.EELS_MnFe()
s.plot()
```





Select navigation `pixel 5` and invert signal axis:

```
s.inav[5].isig[::-1].plot()
```

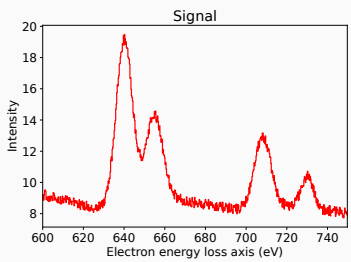


Plot single spectrum signal range: `pixels 600-750`

```
s.inav[5].isig[600:750].plot()
```

Plot single spectrum signal range: `600-750 eV`

```
s.inav[5].isig[600.:750.].plot()
```



What is eXSpy

- HyperSpy extension for EDS and EELS data
- The scientific domains at the roots of HyperSpy
- Adds signal-specific functions
- One of the most powerful HyperSpy extensions

eXSpy features include

- Specific signal types (python classes) for EDS and EELS
- Dedicated models to fit EDS and EELS spectra
- Database of X-ray energies
- EDS quantification (STEM only, not SEM)
- EELS: Supports different datasets for the Generalised Oscillator Strengths (GOS)
- EELS fine structure analysis

eXSpy

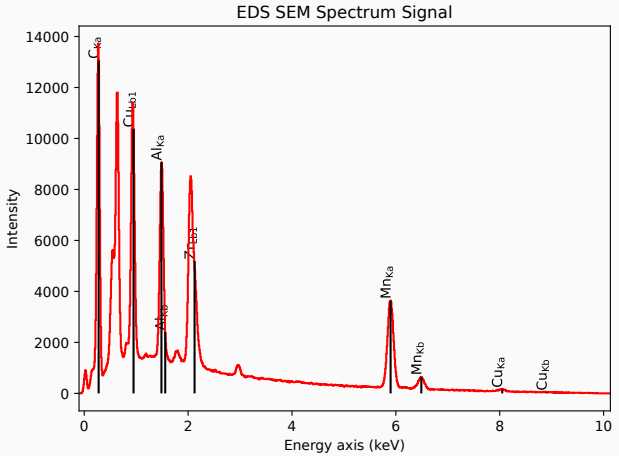
eXSpy features include

- Specific signal types (python classes) for EDS and EELS
- Dedicated models to fit EDS and EELS spectra
- Database of X-ray energies
- EDS quantification (STEM only, not SEM)
- EELS: Supports different datasets for the Generalised Oscillator Strengths (GOS)
- EELS fine structure analysis
- All of the built-in HyperSpy tools!

eXSpy

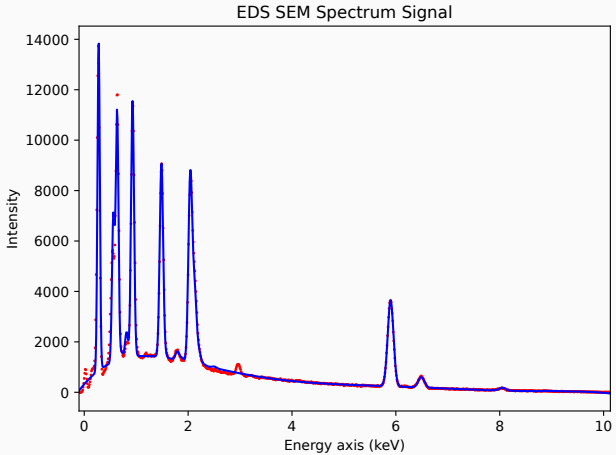
eXSpy examples: Labeling elements in plots

```
s = exspy.data.EDS_SEM_TM002()
s.add_elements(['C','Mn','Cu','Al','Zr'])
s.axes_manager[0].name = 'Energy'
s.plot(True, only_lines=['Ka','b'])
```



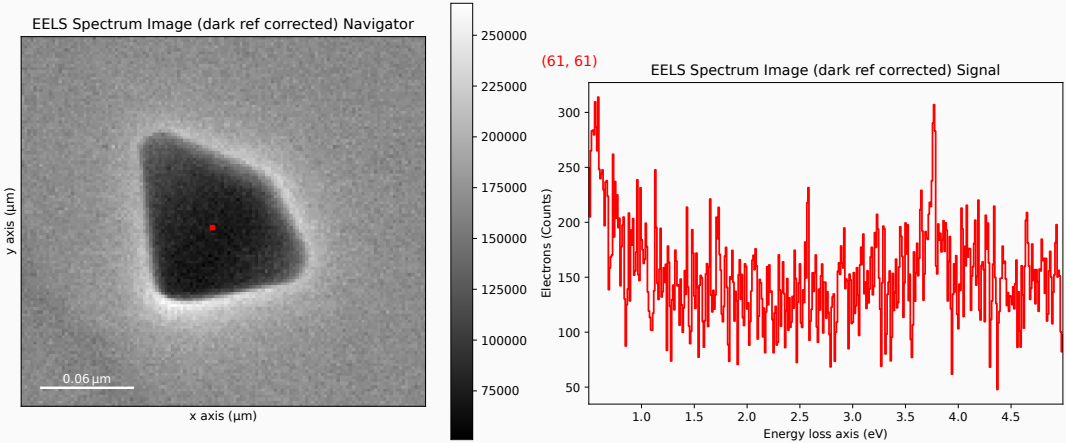
eXSpy examples: Using predefined model


```
m = s.create_model()
m.fit()
m.fit_background()
m.plot()
```



eXSpy examples: Signal decomposition using machine learning

Noisy, low-intensity EELS map:

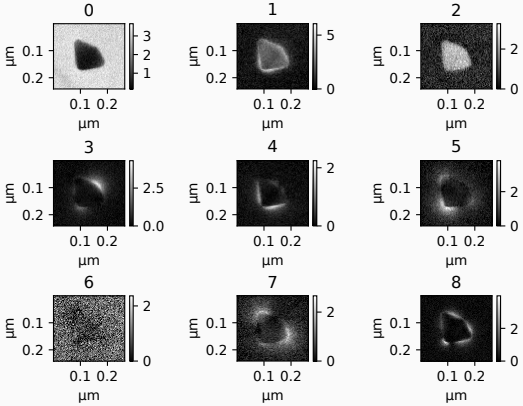


eXSpy examples: Signal decomposition using machine learning

e.g. Non-negative Matrix Factorisation (NMF): identify signal components and distribution

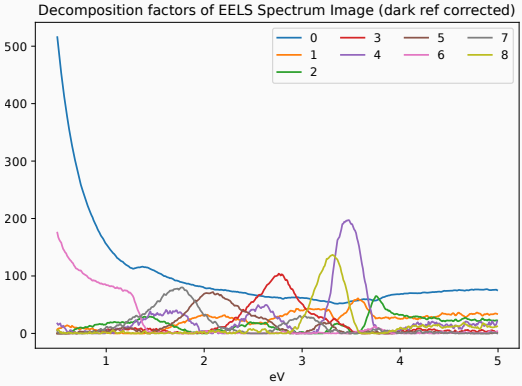
```
s.decomposition(normalize_poissonian_noise=True,algorithm='NMF',output_dimension=9)
s.plot_decomposition_loadings()
```

Decomposition loadings of EELS Spectrum Image (dark ref corrected)




```
s.plot_decomposition_factors()
```

Decomposition factors of EELS Spectrum Image (dark ref corrected)



What is LumiSpy

- `HyperSpy` extension for luminescence spectroscopy data
- Provides the framework to work with photoluminescence (PL), cathodoluminescence (CL), electroluminescence (EL) and Raman spectroscopy
- Adds signal-specific functions
- Work in progress … open for other contributors

λumiSpy

LumiSpy features include

- Specific signal types (python classes) for luminescence spectra and transients
- Transformation of wavelength to electron volt and wavenumbers/Raman shift (non-uniform signal axes)
- Tools such as data normalization/scaling, centre of mass
- Utility functions useful in luminescence spectroscopy data analysis: joining multiple spectra, unit conversion etc.

LumiSpy features include

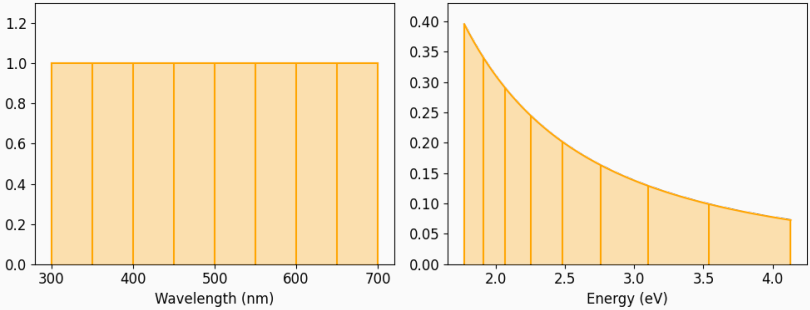
- Specific signal types (python classes) for luminescence spectra and transients
- Transformation of wavelength to electron volt and wavenumbers/Raman shift (non-uniform signal axes)
- Tools such as data normalization/scaling, centre of mass
- Utility functions useful in luminescence spectroscopy data analysis: joining multiple spectra, unit conversion etc.
- All of the built-in HyperSpy tools!

Jacobian Transformation: Wavelength → Energy axis

Preserve integral under function $I(E)dE = I(\lambda)d\lambda$

after non-uniform axis transformation $E = hc/\lambda$

$$I(E) = I(\lambda)\frac{d\lambda}{dE} = I(\lambda)\frac{d}{dE}\frac{hc}{E} = -I(\lambda)\frac{hc}{E^2}$$



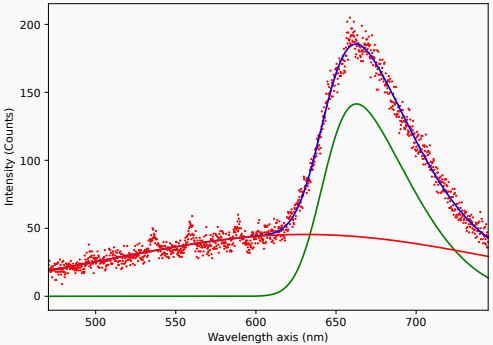

```
s.to_eV()
```

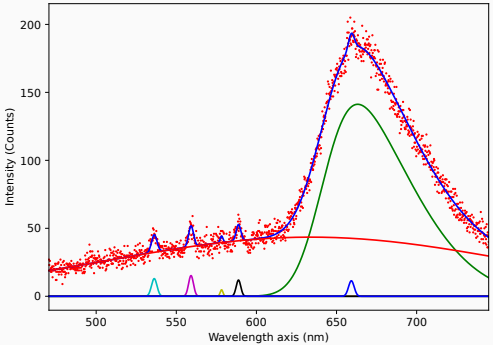
[J. Mooney and P. Kambhampati, The Journal of Physical Chemistry Letters 4, 3316 (2013). doi:10.1021/jz401508t]
[Y. Wang and P. D. Townsend, J. Luminesc. 142, 202 (2013). doi:10.1016/j.jlumin.2013.03.052]

LumiSpy examples: Scripted fitting of a complex model

CL spectrum of luminescent centers in h-BN on SiO_2

- SiO_2 emission: broad Gaussian & skew normal distribution
- Automatically identify sharp peaks in spectrum and add additional Gaussian peaks



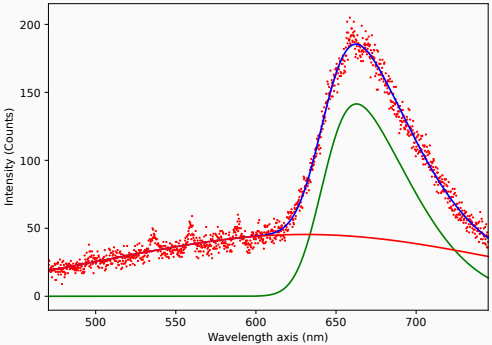


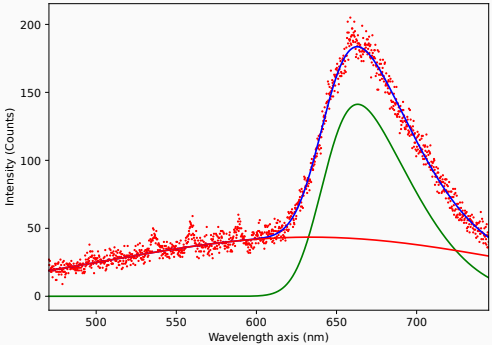
[Hernández-Mínguez et al., Phys. Rev. Appl. 10, 2331 (2018)]

LumiSpy examples: Scripted fitting of a complex model

CL spectrum of luminescent centers in h-BN on SiO_2

- SiO_2 emission: broad Gaussian & skew normal distribution
- Automatically identify sharp peaks in spectrum and add additional Gaussian peaks
- Yields refined background spectrum



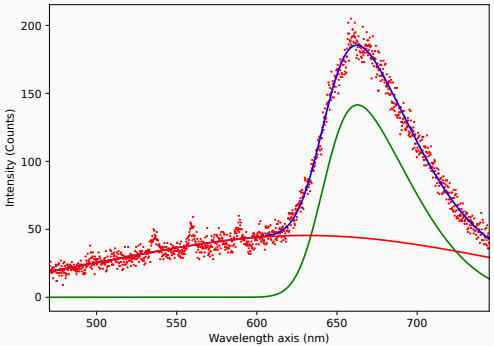


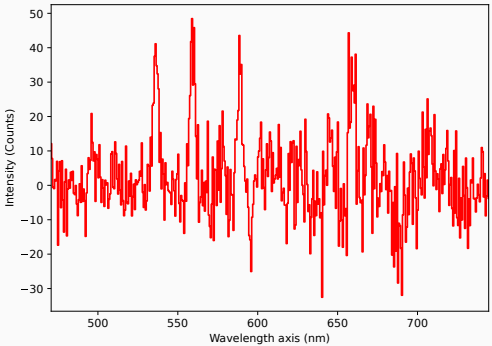
[Hernández-Mínguez et al., Phys. Rev. Appl. 10, 2331 (2018)]

LumiSpy examples: Scripted fitting of a complex model

CL spectrum of luminescent centers in h-BN on SiO_2

- SiO_2 emission: broad Gaussian & skew normal distribution
- Automatically identify sharp peaks in spectrum and add additional Gaussian peaks
- Yields refined background spectrum, which can then be subtracted

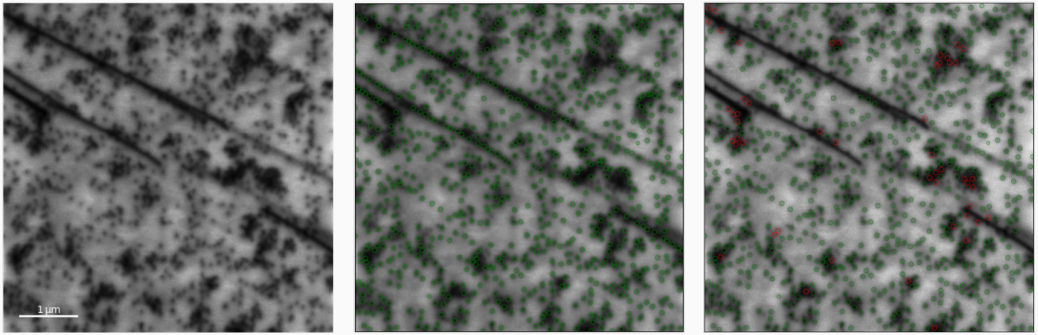




[Hernández-Mínguez et al., Phys. Rev. Appl. 10, 2331 (2018)]

LumiSpy examples: Dislocation density in (In,Ga)N layers

- Panchromatic CL maps show dislocations as dark spots
- Automated spot identification using Laplacian of Gaussian algorithm
- Manual optimization

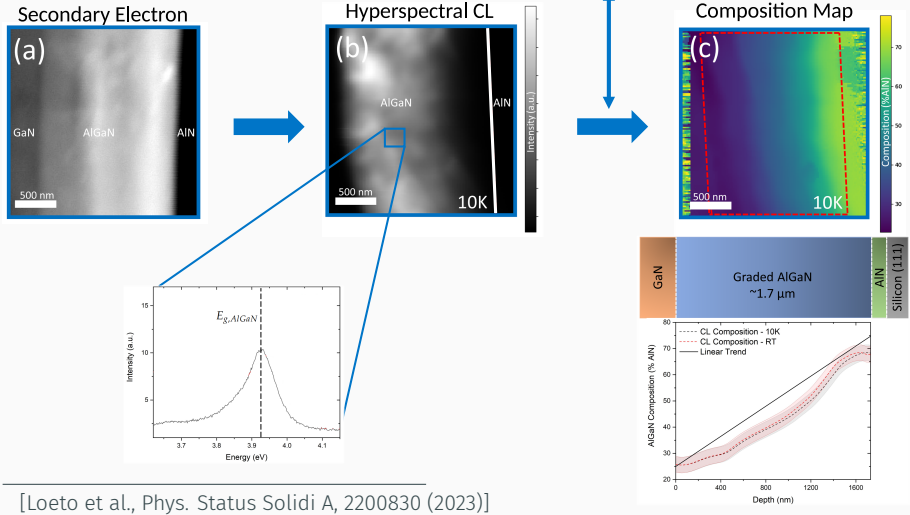


$N_{TD} = 860, \rho_{TD} = 2.56 \times 10^9$ cm^{-2}

[Campbell, Master thesis, Strathclyde Univ. (2024)]

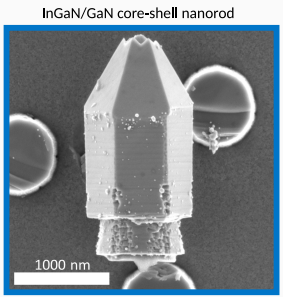
LumiSpy examples: Quantifying alloy composition of graded (Al,Ga)N buffer

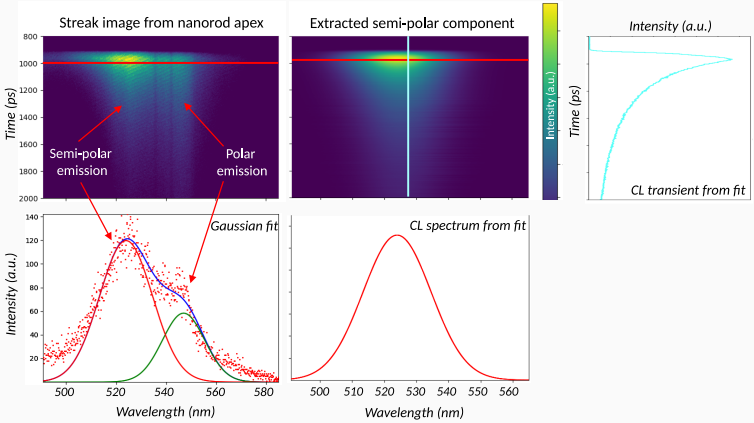
$$E_{g,AlGaN} = xE_{g,AlN} + (1-x)E_{g,GaN} - bx(1-x)$$



[Loeto et al., Phys. Status Solidi A, 2200830 (2023)]

LumiSpy examples: Separating emission bands from streak-camera data





[Loeto et al., Nanotechnology, under review (2024)]

The community: Resources – getting started and obtaining help

HyperSpy

- Website: https://hyperspy.org
- User guide: https://hyperspy.org/hyperspy-doc/current
- Tutorial notebooks: https://github.com/hyperspy/hyperspy-demos
- Gitter chat: https://gitter.im/hyperspy/hyperspy

LumiSpy

- User guide: https://lumispy.org
- Tutorial notebooks: https://github.com/LumiSpy/lumispy-demos

eXSpy

- User guide: https://hyperspy.org/exspy
- Tutorial notebooks: https://github.com/hyperspy/exspy-demos

The community: Contributing

Anyone can make LumiSpy/eXSpy/HyperSpy better! Don't be shy!

- Community-driven development
- Report bugs on the github issue tracker
- Improve documentation
- Help review pull requests on github
- Contribute new functionalities:
  - Include tests and documentation
  - More effort than scripting for personal use
  - \+ Shared development (avoid reinventing the wheel)

Cite HyperSpy and its extensions

HyperSpy, LumiSpy and eXSpy have DOIs provided by Zenodo. There is a generic DOI for the whole project, but every version automatically gets a dedicated DOI assigned.

Getting started for Tutorials

Interactive tutorial presented as Jupyter Notebook – Follow on your own Laptop!

You will be grouped based on prior knowledge (self-assessment).

1. Install python and relevant libraries on your Laptop:

- We recommend installing the `HyperSpy Bundle`
- If you already have python installed, get the following packages from `pip` or `conda`:
  - `hyperspy, lumispy, exspy, hyperspy-gui-ipywidgets, scikit-learn`
- See also Installation Guides at https://lumispy.org and https://hyperspy.org
- A number of colleagues here can help you in the coffee breaks to get started!

2. Get the demo notebooks and data:

- https://github.com/lumispy/eBEAM2024-Tutorial

PS: Find this presentation and relevant links in the above github repository!