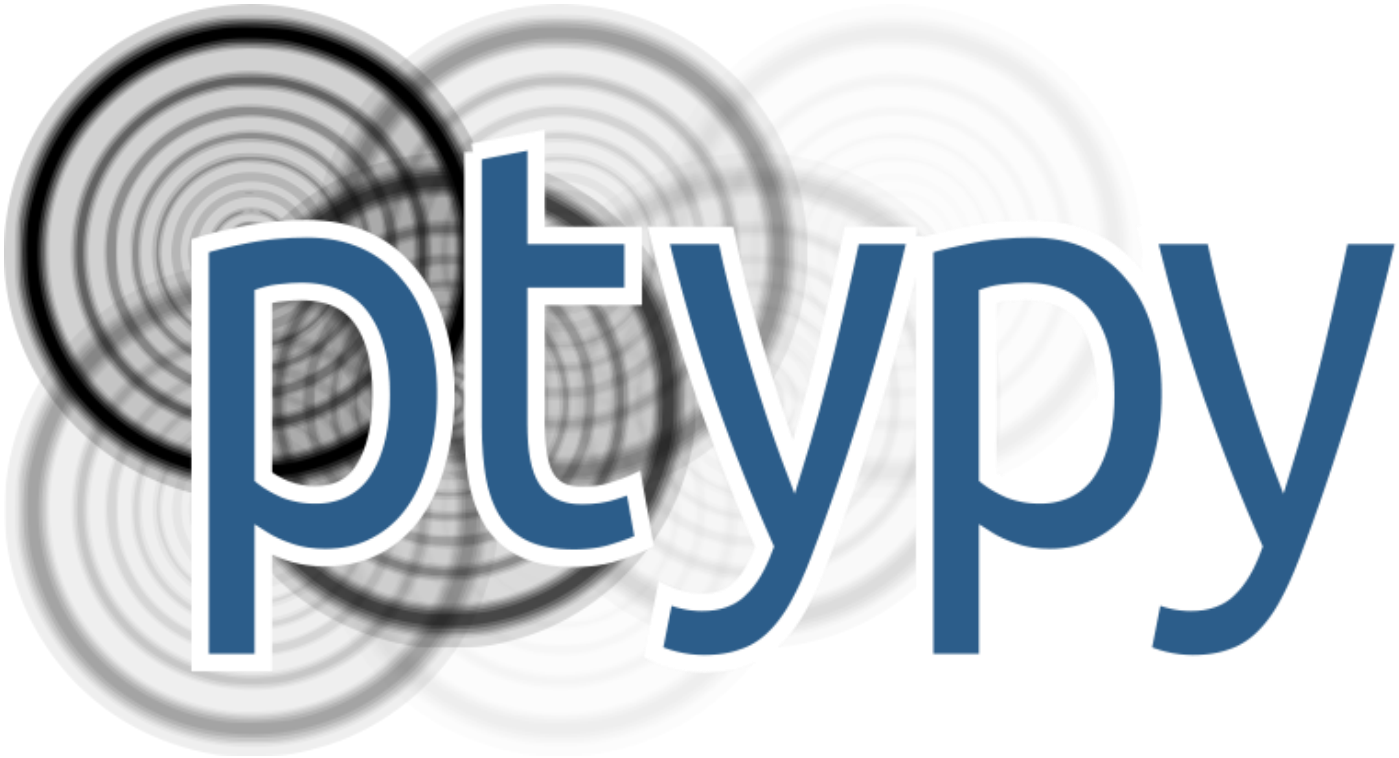


Benedikt Daurer

An introduction to PtyPy

PtyPy 2025 @SOLEIL, 12th and 13th May 2025

What is PtyPy – here are the facts

- PtyPy is a **framework** for scientific **ptychography**
- PtyPy is a **community** project: https://github.com/ptycho/ptypy
- License: academic / non-commercial
- The framework is written in **Python**
- **GPU support**: optimised CUDA kernels wrapped with CuPy/PyCUDA
- **Algorithms:** DM, ML, RAAR, ePIE, sDR
- **Features**: MPI parallelisation, mixed states, position-refinement, on-the-fly reconstructions, client-server visualisaton
- **Tutorials**: https://ptycho.github.io/tutorials
- **Citation**: B.Enders and P.Thibault, *Proc. R. Soc. A* **472**, doi

The History of PtyPy

- **2007 – 2009**
  early versions of the software were developed by Pierre Thibault at Paul Scherrer Institut (PSI) in Matlab
- **2009 – 2010**
  Pierre relocated to Technical University Munich (TUM) and joined E17 group (Franz Pfeiffer) – code rewritten in Python
  supported features: DM, ML, parallel, modes
- **2011 – 2012**
  The ptycho team at TUM grows, contributions from Martin Dierolf, Björn Enders and Marco Stockmar
  added features: nearfield, spectral modes
- at this time many people wanted the code but it was a mess!

The History of PtyPy

- **2013 – 2014**
  Complete redesign from scratch by Pierre and Björn, focusing on data structures, now using git for version control
- **October 2014**: 0.1.0 release for XRM conference in Melbourne
- **2015**: PtyPy becomes a private repository
- **2016**: PtyPy paper is published
- **October 2018**: 0.3.0 release with major changes and contributions from Pierre, Björn, Alexander Björling and Aaron Parsons
  added features: Bragg ptychography

The History of PtyPy

- **November 2019**: 0.4.0 release with change to Python 3 only and support for position refinement, contributions from Björn, Alex, Aaron, Julio Cesar Da Silva, Wilhelm Eschen and Benedikt Daurer
- **May 2022**: 0.5.0 release with major upgrade on high-performance using GPU acceleration, supported by Diamond Light Source with contributions from Björn, Aaron, Benedikt and Jörg Lotze (Xcelerit) added features: ePIE, RAAR, sDR
- **December 2022**: 0.6.0 release and PtyPy is public again under new license
- **January 2023**: 0.7.0 release together with tutorials in preparation for our first PtyPy workshop

The History of PtyPy

- **February 2024:** 0.8.0 release with new Cupy engines, WASP and ThreePIE as custom engines
- **June 2024:** Updated tutorials with most datasets now publicly available
- **May 2025**: 0.9.0 release with the wavefield preconditioner added to all ML engines

List of PtyPy contributors

- Pierre Thibault
- Björn Enders
- Martin Dierolf
- Marco Stockmar
- Alexander Björling
- Aaron Parsons
- Julio Cesar Da Silva
- Wilhelm Eschen
- L. Bloch
- Benedikt Daurer
- Simone Sala
- Stefanos Chalkidis
- Jörg Lotze
- Susanna Hammarberg
- Timothy Poon
- Jaroslav Fowkes
- and more…

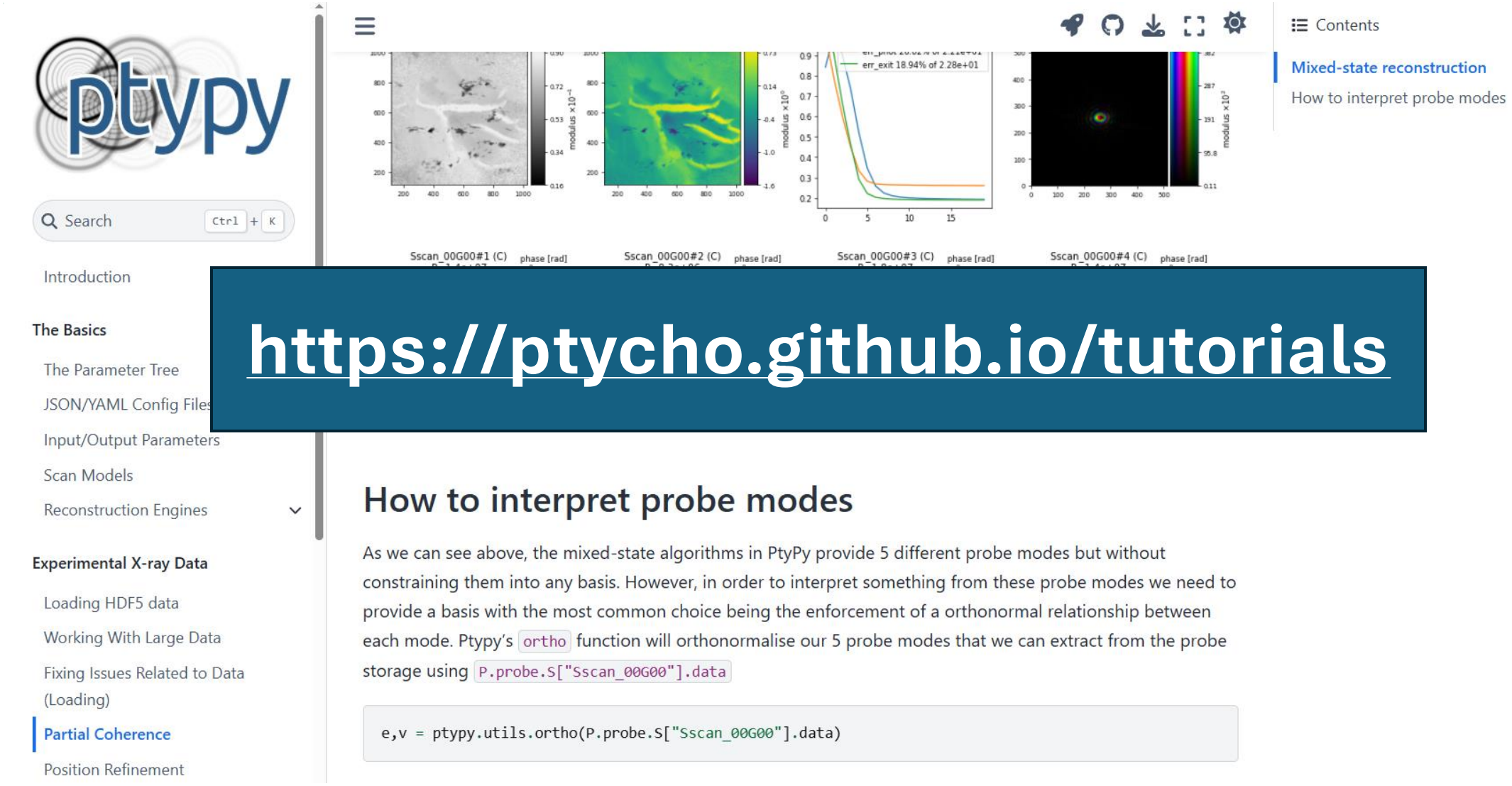
Current PtyPy development team

Björn Enders
(NERSC, US)

Benedikt Daurer
(Diamond Light Source, UK)

Pierre Thibault
(University of Trieste, Italy)

The online tutorials



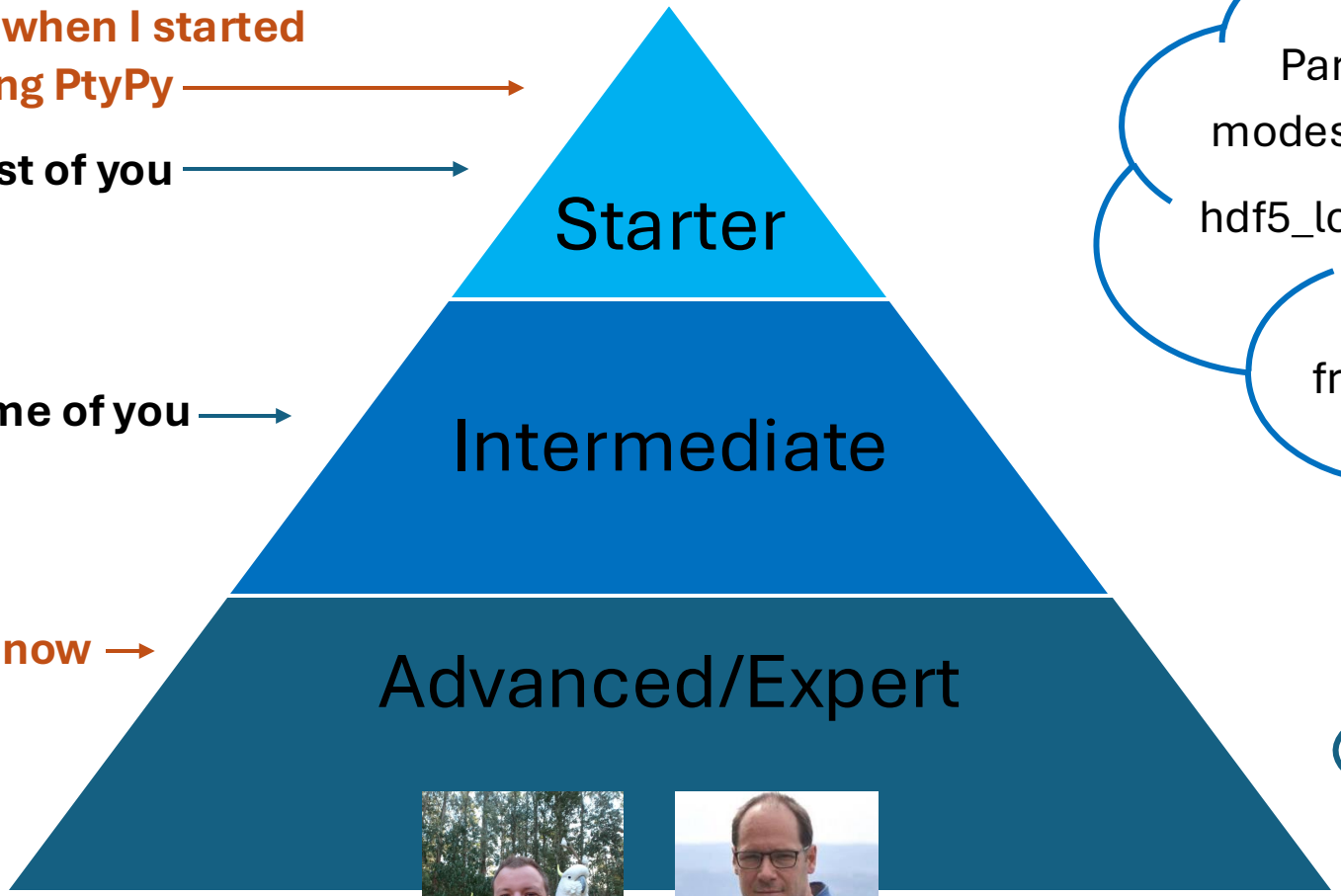
PtyPy/Ptychography at different levels

Me when I started using PtyPy →

Most of you →

Some of you →

Me now →



THIS WORKSHOP

Parameter tree
geometry
engines
modes
scans
P=Ptycho(level=5)
hdf5_loader
out-of-memory
fourier_power_bound
Multi-GPU
frames_per_block
PODs
subclass

CUDA kernels
Fix bugs
New features

DEV HACKATHONS

PtyPy for Starters

Basic Python installation

```
$ git clone https://github.com/ptycho/ptypy.git
$ cd ptypy
$ pip install .
```

Full installation with conda

```
$ conda env create -f dependencies_full.yml
$ conda activate ptypy_full
(ptypy_full)$ pip install .
```

Full installation with conda and GPU support

```
$ conda env create -f ptypy/accelerate/cuda_cupy/dependencies.yml
$ conda activate ptypy_cupy
(ptypy_cupy)$ pip install .
```

PtyPy for Starters

Parameters defined like a dictionary in Python

```
# Create parameter tree
p = u.Param()
```

Top level parameters

```
# Set verbose level, can be "interactive", "info" or "debug“
p.verbose_level = "interactive"
```

I/O parameters

```
# Basic I/O settings (no files saved in this case)
p.io = u.Param()
p.io.rfile = None
p.io.autosave = u.Param(active=False)
p.io.autoplot = u.Param(active=False)
p.io.interaction = u.Param(active=False)
```

Scan Parameters

```
# Define the scan model
p.scans = u.Param()
p.scans.MF = u.Param()
p.scans.MF.name = "Full"
```

- **Hands-on tutorials I: Understanding the PtyPy parameter tree**

PtyPy for Starters

Data Parameters

```
# Data loader / simulator
p.scans.MF.data= u.Param()
p.scans.MF.data.name = "MoonFlowerScan“
p.scans.MF.data.shape = 128
p.scans.MF.data.photons = 1e8
p.scans.MF.data.num_frames = 200
p.scans.MF.data.density = 0.2
p.scans.MF.data.psf = 0.
```

Engine Parameters

```
# Define reconstruction engine
p.engines = u.Param()
p.engines.engine00 = u.Param()
p.engines.engine00.name = "DM“
p.engines.engine00.numiter = 80
```

The run command

```
# Prepare and run
P = ptypy.core.Ptycho(p,level=5)
```

- **Hands-on tutorials I: Understanding the PtyPy parameter tree**

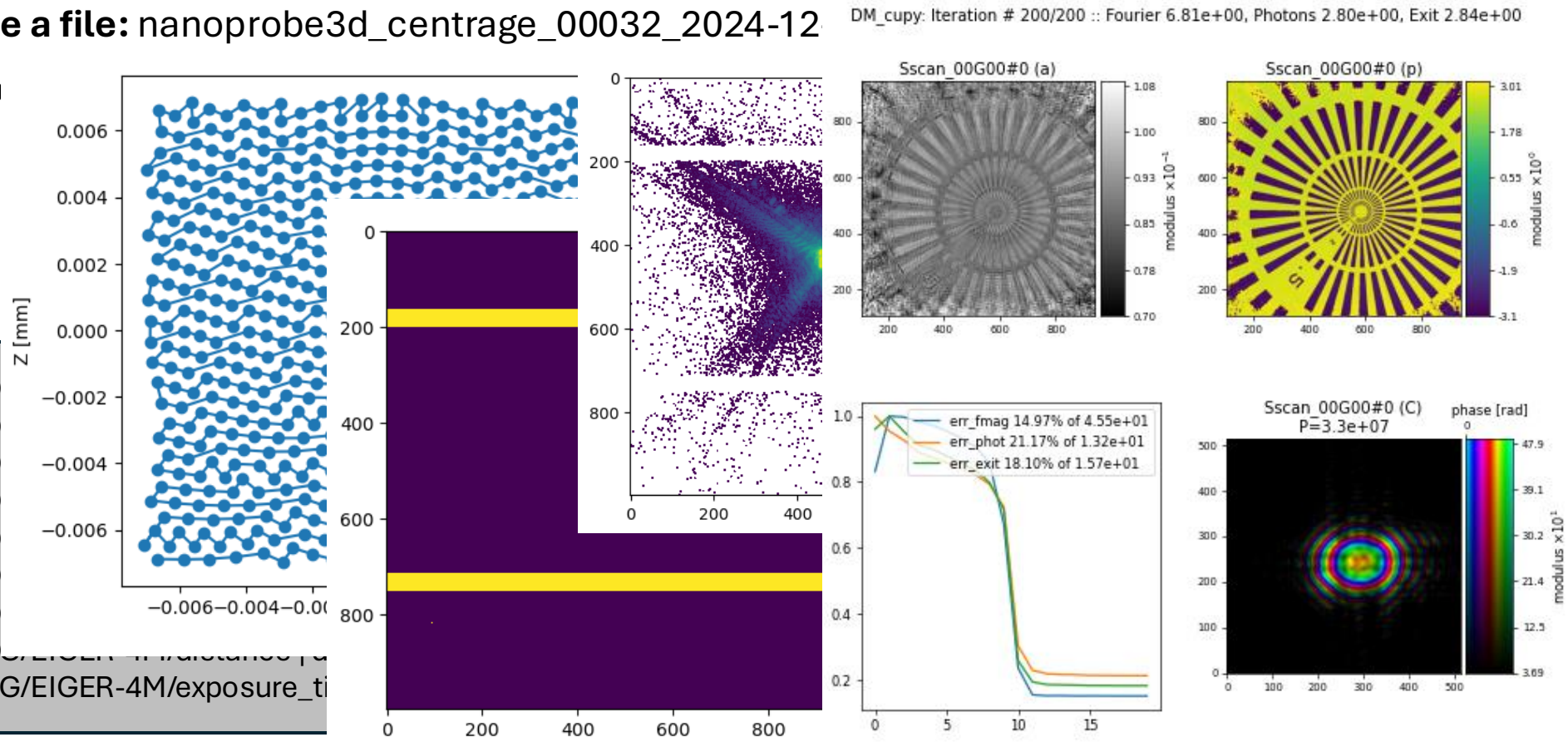
PtyPy for Starters

Common scenario: “Can you please reconstruct this dataset for me?”

Javier gave me a file: nanoprobe3d_centrage_00032_2024-12

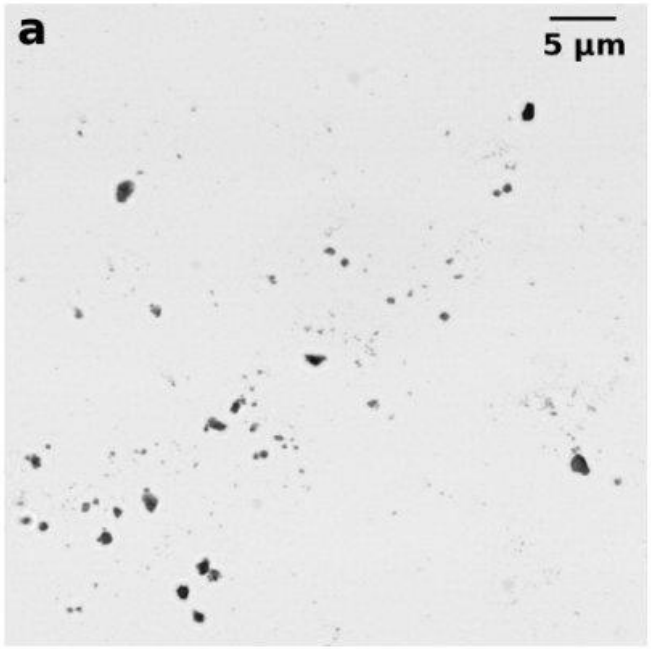
He also adde
I am available
Best regards,
Javier

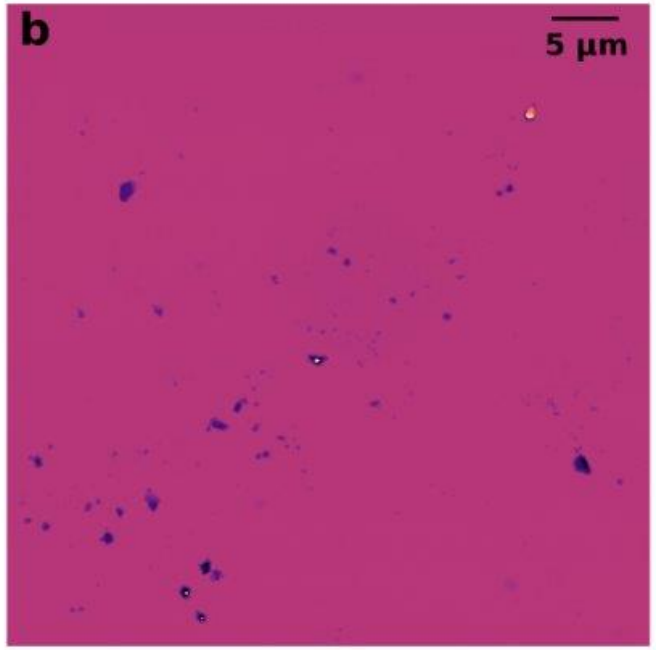
/SiemensStar_00032/SWIN
/SiemensStar_00032/SWIN
/SiemensStar_00032/SWIN
/SiemensStar_00032/SWIN
/SiemensStar_00032/SWIN
/SiemensStar_00032/SWIN
/SiemensStar_00032/SWIN
/SiemensStar_00032/SWIN
/SiemensStar_00032/SWING/EIGER-4M/exposure_ti

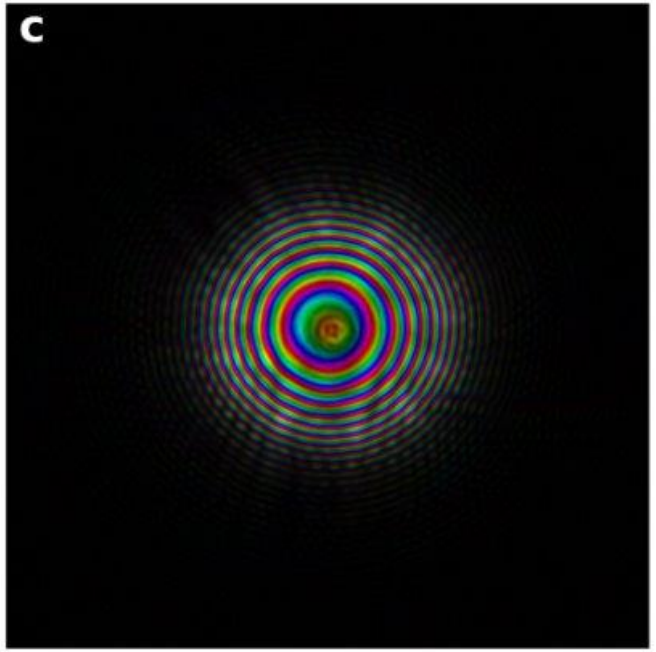


- **Hands-on tutorials II: Working with experimental data**

PtyPy for the Starter/Intermediate







Benchmarks
4 x A100 GPUs

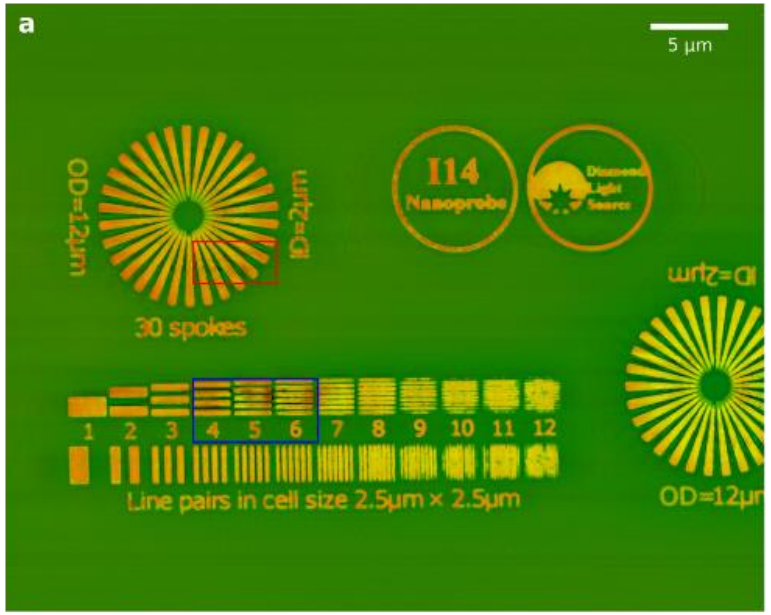
Data loading: 38 s
Prepare engine: 37 s
Iterations: 202 s
Total: 277 s

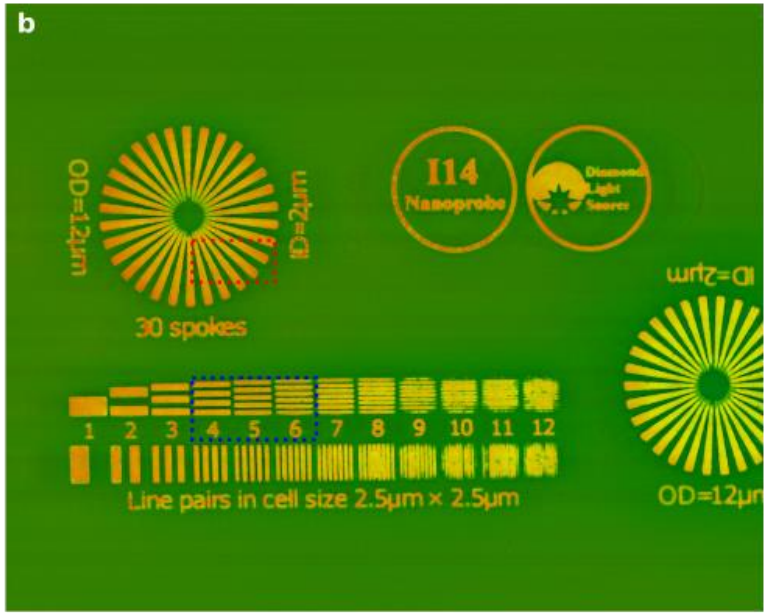
0.2 nanoseconds / pixel / iteration

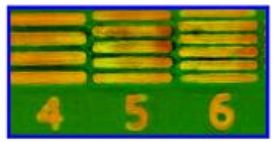
Fe nanoparticle dataset collected at I08-1 with 4225 diffraction patterns of size 512x512 with the reconstructed object amplitude (a) and phase (b) shown together with a single-mode probe (c) after 1000 iterations of DM.

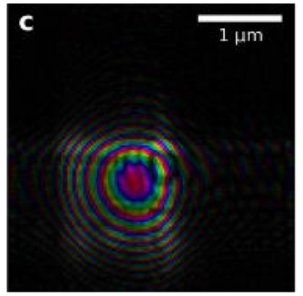
- **Hands-on tutorials III: Working with large data sets**

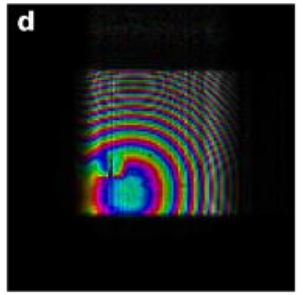
PtyPy for the Starter/Intermediate











<u>w/o position refinement</u>
Data loading: 238 s
Prepare engine: 77 s
Iterations: 112 s
Total: 427 s

<u>with position refinement</u>
Data loading: 231 s
Prepare engine: 75 s
Iterations: 115 s
Total: 421 s

0.1 nanoseconds / pixel / iteration

Large I14 test dataset with 200 000 diffraction patterns of size 128x128 reconstructed using 300 iterations of DM without (a) and with (b) position refinement. There is a clear improvement in contrast at no significant extra computational cost. The reconstructed single-mode probe (c) and its Fourier transform (d) are shown on the right.

- **Hands-on tutorials III: Working with large data sets**

PtyPy for the Intermediate

ptypy/ptypy/experiment/cSAXS.py

```
21
22      @register()
23      class cSAXSScan(PtyScan):
24          def __init__(self, pars=None, **kwargs):
25              '''
26              Defaults:
27              [name]
28              default = cSAXS
29              type = str
30              help = the reference name of this loader
31              doc =
32
33              [base_path]
34              default = None
35              type = str
36              help = the base path for reconstruction
37              doc =
38
39              [visit]
40              default = None
41              type = str
42              help = the visit number
43              doc =
```

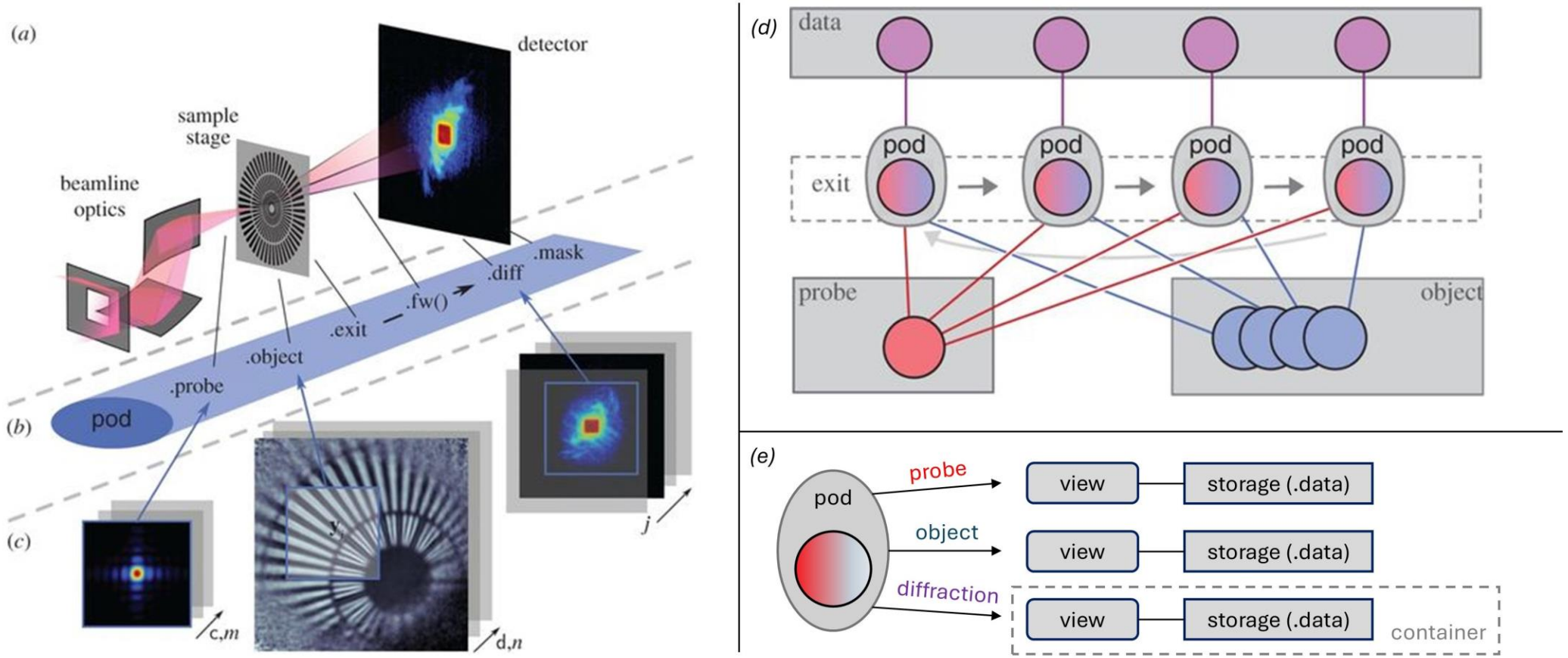
ptypy/ptypy/custom/WASP.py

```
23
24      @register()
25      class WASP(base.PositionCorrectionEngine):
26          """
27          Weighted Average of Sequential Projections
28
29          Defaults:
30
31          [name]
32          default = WASP
33          type = str
34          help =
35          doc =
```

Thanks to **Timothy Poon** for the implementation of the WASP engine!

- **Hands-on tutorials IV: Write your own data loader**
- **Hands-on tutorials IV: Write your own reconstruction engine**

PtyPy for the Intermediate



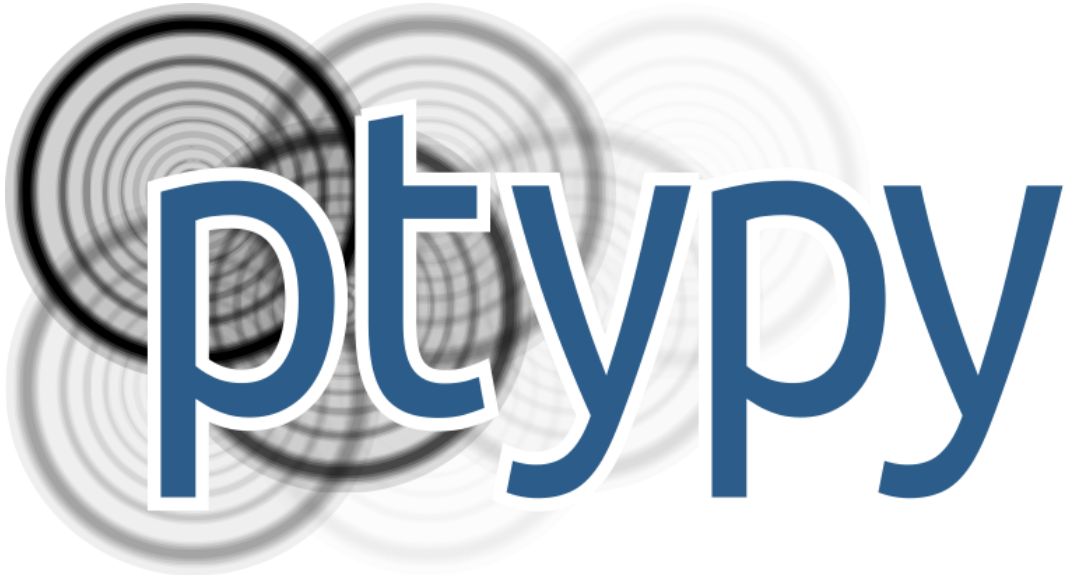
- **Basics of Ptychography I and II**

PtyPy for the Advanced

Out of scope for this workshop, but the next hackathon is just around the corner

Trieste, 15-19 September 2025

Speak to us if you are interested to join



https://ptypy.org