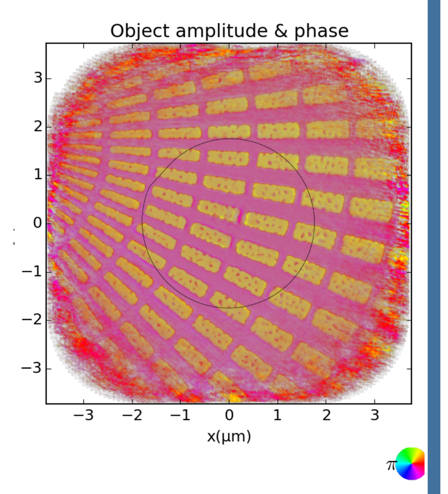
PyNX: a Coherent Imaging toolkit based on Operators and GPU computing

V. Favre-Nicolin[1,2,@]
[1] ESRF- The European Synchrotron, Grenoble,France [2] Univ. Grenoble Alpes, INAC-SP2M, Grenoble, France
@ favre@esrf.eu

Coherent Diffraction Imaging experiments will become much more intense/faster after the EBS upgrade. The ESRF has started an open-source scientific software programme, to improve the ability of users to quickly analyse data for the various techniques, notably on the coherent imaging beamlines – to encourage users interested primarily in the results on materials rather than in the development of the technique. The PyNX development is based on Python3, and fully exploits GPUs for high-performance computing, with the OpenCL and CUDA languages.

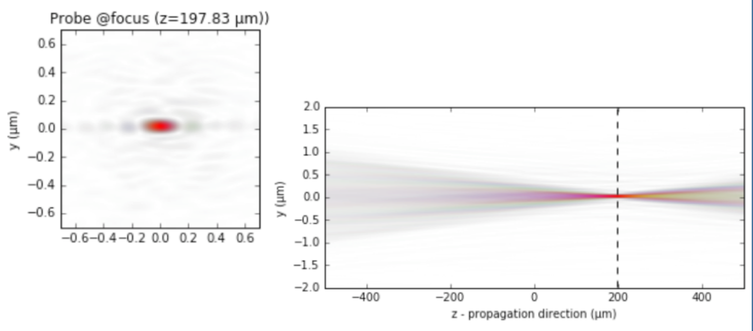
Main PyNX features

All algorithms fully executed on GPU for performance
Use *normalized log-likelihood* as a systematic figure of merit



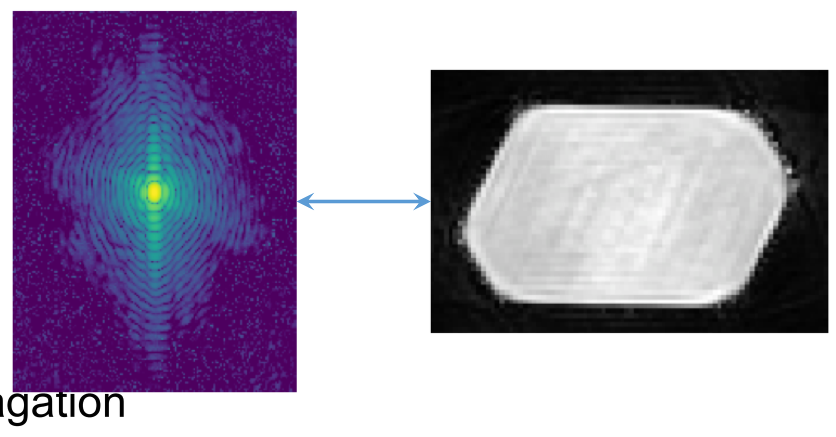
2D Ptychography (far field & near field, Bragg):

- **Combination of algorithms:**
  - Difference Map
  - Alternating projections
  - Conjugate Gradient Maximum Likelihood
  - Incoherent background
- **Example performance** (1 x nVidia V100 GPU):
  - 2D Ptycho on 142 frames (700x700 pixels):
    - 0.13s/cycle (Difference map)
    - 0.27s/cycle (Maximum likelihood)
- **Use and create CXI format datasets**
  - http://cxidb.org/cxi.html



Coherent Diffraction Imaging:

- **Algorithms (2D, 3D)**:
  - HIO, ER, CF…
  - Maximum Likelihood
  - Partial coherence
  - Likelihood-sorting of solutions



Wavefront propagation:

- Near field propagation
- Far field propagation
- Fractional Fourier transform propagation

Scattering from atomistic models:

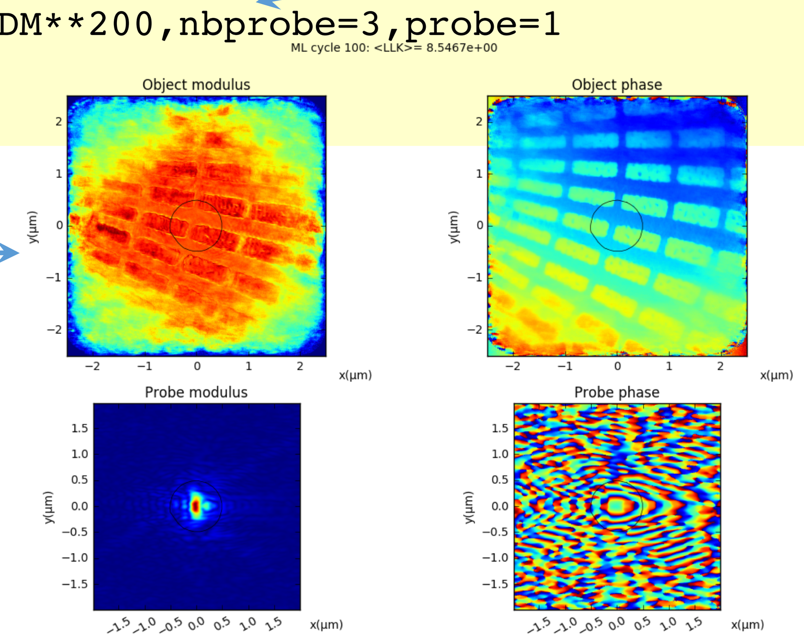
- Fast calculations using CUDA or OpenCL
- 7x10^11 atoms.reflections/s using a single *Titan V* GPU (~5 Tflop/s)

Analyse your Ptycho data @ESRF

Command-line script (beamline GPU workstations)

```
pynx-id01pty.py specfile=align.spec scan=13,14
        imgname=mpx4/data_mpx4_%05d.edf.gz loadmask=maxipix
        probe=60e-6x200e-6,0.09 defocus=100e-6
        detectordistance=1.386
        ptychomotors=pix,piz,-x,y
        algorithm=analysis,ML**100,DM**200,nbprobe=3,probe=1
        saveplot save=all  liveplot
```

Chain algorithms



Use the OAR ESRF Cluster
Deployed in a python virtual environment

PyNX

https://software.epn-campus.eu

This poster

Projects

- **More applications to (ESRF) beamlines**
- **Ptychography**:
  - Optimisation of scan positions
  - Bragg (3D, back-projection) (collab. V. Chamard, M. Allain)
  - More projection algorithms
  - Pink beam
- **CDI**:
  - Combining solutions, genetic algorithms
  - Bragg using known probe (multiple modes,..)
  - Improve recovery of signal behind beamstop

Fast coherent X-ray imaging Using Operators, Python & GPUs

All coherent imaging algorithms are based on simple combinations of mathematical operations:

- **Near and far field propagation**
- **Applying constraints in real and Fourier space**

TABLE I. Summary of various algorithms.

Algorithm	Iteration $\rho^{(n+1)}=$
ER	$P_s P_m \rho^{(n)}$
SF	$R_s P_m \rho^{(n)}$
HIO	$\begin{cases} P_m \rho^{(n)}(r) & r \in S \\ (I - \beta P_m)\rho^{(n)}(r) & r \notin S \end{cases}$
DM	$\{I + \beta P_s[(1+\gamma_s)P_m - \gamma_s I] - \beta P_m[(1+\gamma_m)P_s - \gamma_m I]\}\rho^{(n)}$
ASR	$\frac{1}{2}[R_s R_m + I]\rho^{(n)}$
HPR	$\frac{1}{2}[R_s(R_m + (\beta-1)P_m) + I + (1-\beta)P_m]\rho^{(n)}$
RAAR	$[\frac{1}{2}\beta(R_s R_m + I) + (1-\beta)P_m]\rho^{(n)}$

Marchesini, S. '*A unified evaluation of iterative projection algorithms for phase retrieval*'. Review of Scientific Instruments **78** (2007), 011301

All PyNX modules are based on operators:

- Called from Python
- Executed on GPU using OpenCL or CUDA
- GPU complexity is completely hidden
- Allowing simple mathematical combinations

CDI operators:

- ER(): error reduction
- HIO(): hybrid input-output
- SupportUpdate()

Using an operator-based API allows to chain operations without knowing anything about GPU programming : simple algorithm developments !

Ptychography operators:

- DM(): difference map
- ML(): maximum likelihood

Chaining algorithms with a CDI object with a single python command:

```
cdi = (SupportUpdate() * ER()**20 * HIO()**100)**4 * cdi
```

Chaining algorithms with a Ptycho object with a single python command:

```
p = ML(update_probe=1,update_obj=1)**100 * DM(update_probe=1,update_obj=1)**400 * p
```

```
################## Try again from a loose support #################################################
%matplotlib inline
plt.figure(1, figsize=(8, 8))
support = np.flipud(fabio.open("/home/pynx/PYNX-TUTORIAL/1-PyNX-scripts/CDI/ESRF-logo-2D/mask_logo5mu_20sec.edf").data)
support = gaussian_filter(support.astype(np.float32), 4) > 0.2
cdi = CDI(fftshift(iobs), obj=None, support=fftshift(support), mask=fftshift(mask), wavelength=1e-10,
          pixel_size_detector=55e-6)

# Initial scaling, required by mask
cdi = ScaleObj(method='F') * cdi

# Do 200 cycles of HIO, displaying object after
cdi =  ShowCDI(fig_num=-1) * HIO() ** 200 * cdi

# Compute LLK
IFT() * LLK() * FT() * cdi
print("LLK_n = %8.3f" % (cdi.get_llk(noise='poisson')))

for i in range(20):
    # Support update operator
    s = 0.5 + 2 * np.exp(-i / 4)
    sup = SupportUpdate(threshold_relative=0.25, smooth_width=s, force_shrink=False, post_expand=(1,-1))

    # Do 40 cycles of RAAR, then 5 of ER
    cdi = ER() ** 5 * HIO() ** 40 * cdi

    # Update support & display current object & diffraction
    cdi = sup * cdi

    IFT() * LLK() * FT() * cdi

    print("HIO+ER #%3d: LLKn = %8.3f" % (i * 45, cdi.get_llk(noise='poisson')))

cdi = ShowCDI(fig_num=1) * cdi
```

Example code for CDI data analysis

```
import h5py as h5
import numpy as np
from pynx.ptycho import *
import pynx.wavefront as wavefront

# Read ptyd data
h5data = h5.File('data/ptyd/sulfided_try1_nfptomo2_subtomo001_0400.ptyd', 'r')
nrj = h5data['/info/energy'].value  # Energy in keV
wavelength = 12.3984e-10 / nrj
detector_distance = h5data['/info/distance'].value
pixel_size_detector = h5data['/info/psize'].value

# sample positions and data
tmp = h5data['/chunks/0/positions'].value
y, x = tmp[:, 0], tmp[:, 1]
iobs = h5data['/chunks/0/data'].value

# Create Ptycho data object
data = PtychoData(iobs=iobs, positions=(y, x), detector_distance=detector_distance, mask=None,
                  pixel_size_detector=pixel_size_detector, wavelength=wavelength, near_field=True)

# Initial probe as a Wavefront object, rectangular aperture
nb_probe = 3
p0 = np.ones((nb_probe, ny, nx), dtype=np.complex64)
# TODO: better define noise, amplitude in secondary modes..
for i in range(1, nb_probe):
    p0[i] = 0.1 * np.random.uniform(0., 1., (ny, nx)) * np.exp(1j * np.random.uniform(0,2*np.pi, (ny, nx)))

# We start from a Wavefront object. Could better start from a propagated one, if optics parameters are known
pr = wavefront.Wavefront(d=p0, pixel_size=pixel_size_detector, wavelength=wavelength)

# Size of the reconstructed object (obj)
nyo, nxo = shape.calc_obj_shape(y / pixel_size_detector, x / pixel_size_detector, (ny, nx))

# Starting object as a phase object
obj0 = np.exp(1j * np.random.uniform(0, 0.5, (nyo, nxo)))

# Main ptycho object
p = Ptycho(probe=pr.get(shift=True), obj=obj0, data=data, background=None)

# Initial scaling
p = ScaleObjProbe() * p

# Optimize
p = DM(update_object=True, update_probe=True, calc_llk=100, show_obj_probe=0) ** 4000 * p
p = ML(update_object=True, update_probe=True, calc_llk=50, show_obj_probe=0) ** 300 * p
```

Example code for near field ptychography data analysis

Acknowledgments for data sets & tests: Steven Leake, Marie-Ingrid Richard, Jerome Carnis, Gaétan Girard (id01), Yuriy Chushkin (id10), Tilman Grünewald, Manfred Burghammer (id13), Julio Cesar da Silva, Leonid Bloch (id16a), Ross Harder (APS)