SRW crash course

What is SRW really good for.

- **Simulating SR wave emission from a charged particle trajectory:**
  - **BM**
  - **IDs**
  - **Arbitrary mag fields**

- **Diffraction effects evaluation**
  - **Mirror sizes**
  - **Apertures**

- **Wavefront propagation on full beamlines***
  - **Full sources complement**
  - **Basic mirror shapes**
  - **Gratings and crystals**
  - **ZPs and CRLs**
  - **Various utilities**

***: Multi electron simulations REQUIRE working on HPC cluster**

What is SRW really NOT good for.

- Quick’n’dirty beamline evaluations

- Power absorption/transmission calculations

- Mirror’s sizing

- Working with “geometrical” sources

Some background basics

- The SRW core was born to calculate BM edge field IR emissions!

(Chubar, Smolyakov, “ Generation of Intensive Long-wavelength Edge Radiation in High-energy Electron Storage Rings”, IEEE 1993)

- That core has been extended to simulate radiation for beam diagnostics

(Chubar, “Precise computation of electron beam radiation in nonuniform magnetic fields as a tool for beam diagnostics”, Review of Scientific Instruments 66, 1872 (1995)

- It was then extended to simulate near-field SR propagation through free space and simple Optical Elements through FFTs.

(Chubar and Elleaume. "Accurate and efficient computation of synchrotron radiation in the near field region." *proc. of the EPAC98 Conference*. Vol. 1177. 1998.

- The core is now a C++ library.

Code at: https://github.com/ochubar/SRW

-It supports bindings on Python

-Ported into Oasys (start from HERE!)

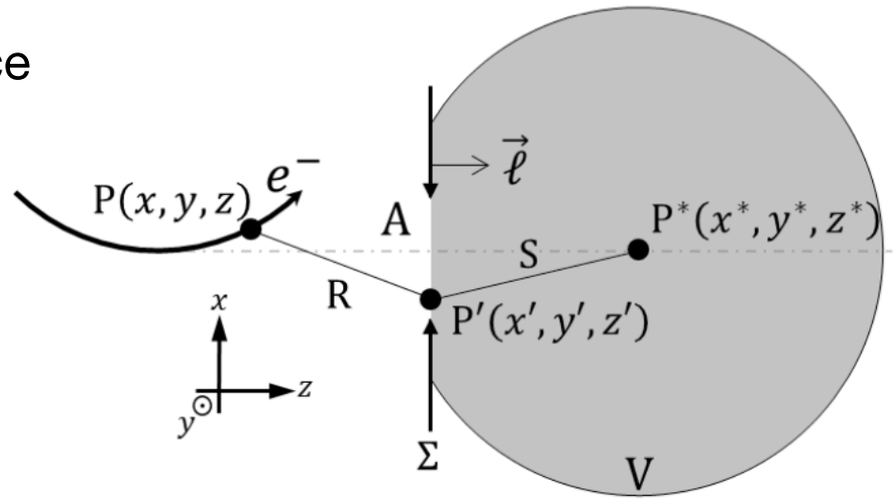
-Best used through oasys-srw

Code at: https://github.com/oasys-kit/OASYS-SRW

Let’s get into it.

Problem:
propagate and arbitrary E field over a distance in free space

Assumptions:
- Prop is in a uniform, uncharged, nonconducting medium
- Prop distance >> wavelength



Huygens - Fresnel

$$U(P^*) = \frac{1}{i\lambda}\iint_A U(P')\frac{\exp(ikS)}{S}\cos\theta\,\mathrm{ds}$$

Fresnel approximation:
a)$z^2 \gg x^*-x'^2+ y^*-y'^2$;
b) binomial expansion of the square root

$$U(P^*) = \frac{e^{ikz}}{i\lambda z}\iint_{-\infty}^{\infty} U(P')\exp\left\{i\frac{k}{2z}[(x^*-x')^2+(y^*-y')^2]\right\}\mathrm{d}x'\mathrm{d}y'$$

Fraunhofer approximation:
a) $z >> \frac{k(x'^2+y'2)_{max}}{2}$

$$U(P^*) = \frac{e^{ikz}e^{i\frac{k}{2z}\left(x^{*2}+y^{*2}\right)}}{i\lambda z}\iint_{-\infty}^{\infty} U(P')\exp\left[-i\frac{2\pi}{\lambda z}(x^*x'+y^*y')\right]\mathrm{d}x'\mathrm{d}y'$$

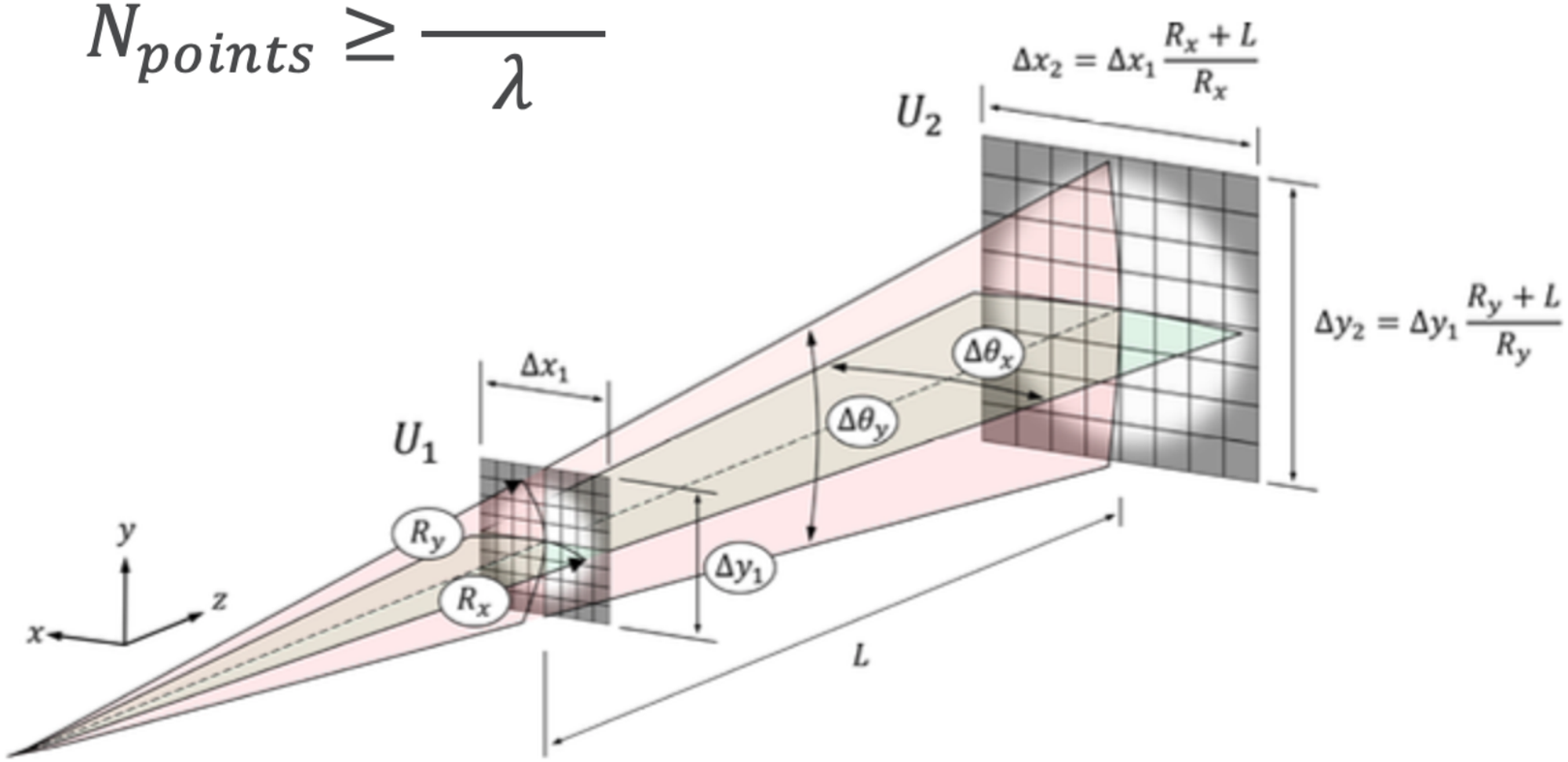
This has to be made discrete :-)

Wavefront and Propagator discretization

$$u_\omega(x,y,z=z^*) = -\frac{ike^{ikz^*}}{2\pi z^*}\sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1} u_\omega(x_i',y_j',z=0)e^{\frac{ik}{2z^*}\left[(x-x_i')^2+(y-y_j')^2\right]}(x_{i+1}'-x_i')(y_{j+1}'-y_j')$$

For each transversal direction:

$$N_{points} \geq \frac{\Delta\theta^2 R}{\lambda}$$

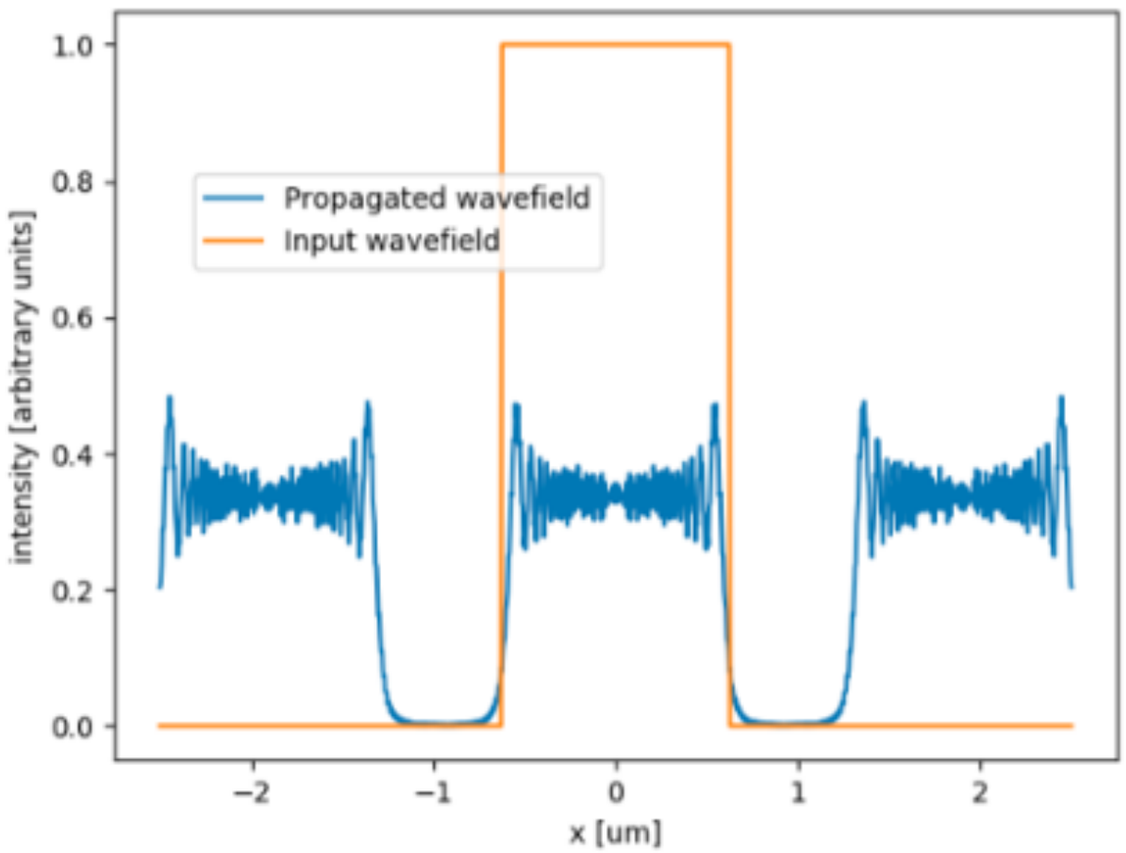


To calculate the disturbance in a grid on the detector plane, the numbers of operation would be $(N_x\ N_y)^2$

Wavefront and Propagator discretization

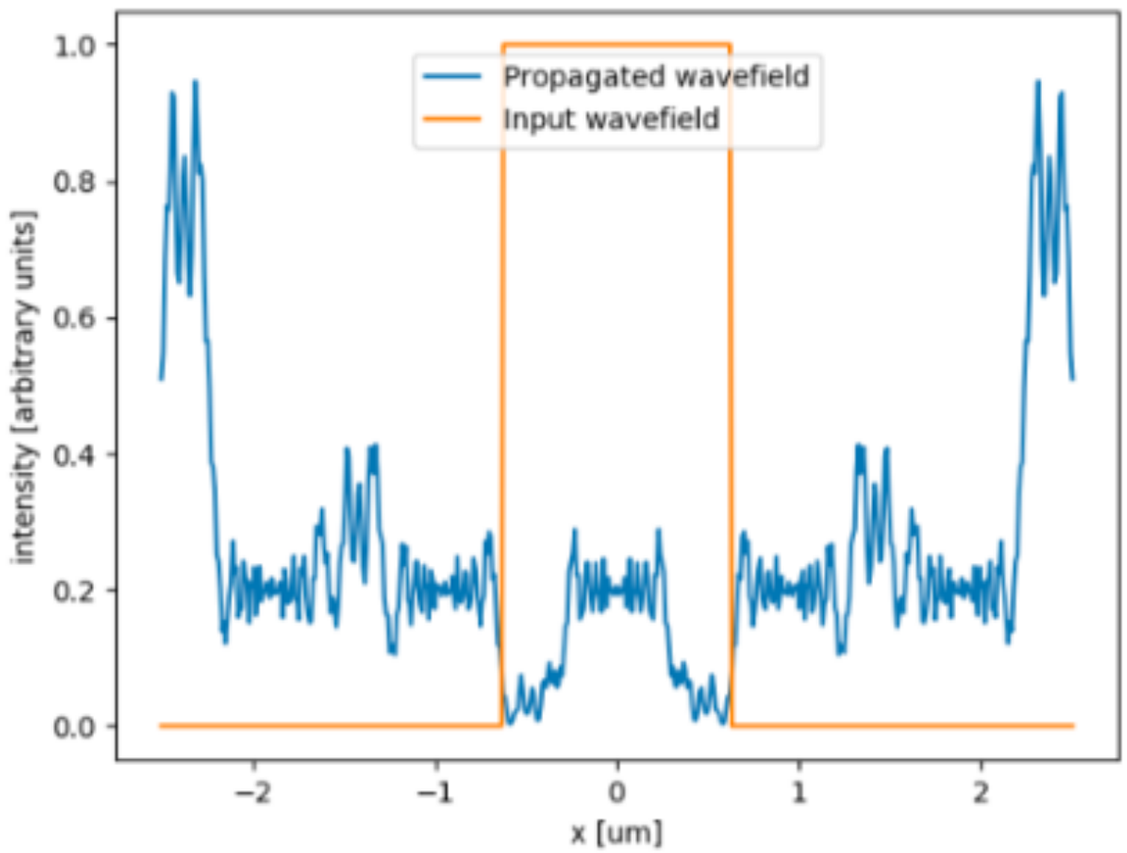
Effect of the discretization: Replicas

Replicas can be avoided by increasing the sampling or by isolating the main feature at the image plane, if the replica are well separated.



Effect of under-sampling: Aliasing

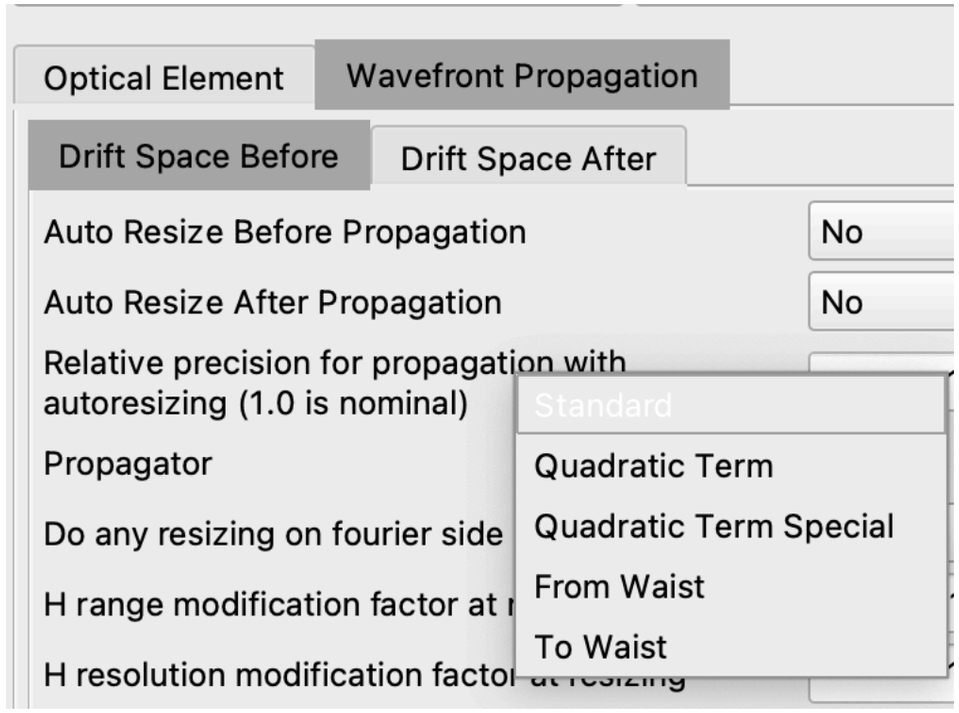
Under-sampling produces an overlapping of the replicas, thus completely distorting the expected result. This effect is called *aliasing.*



SRW propagators

Big source of confusion!!

Must be selected carefully



Difference between propagators: treatment of the phase in the propagation integral

Chubar, O. and Celestre, R., Opt. Express 27, 28750-28759 (2019)

SRW Standard propagator

Is the Fresnel Propagator, uses the convolution theorem, and 2 FFT's.

$$u_\omega(x,y,z=z^*) = -\frac{ike^{ikz^*}}{2\pi z^*}\iint_{-\infty}^{+\infty} u_\omega(x',y',z=0)e^{\frac{ik}{2z^*}\left[(x-x')^2+(y-y')^2\right]}dx'dy' =$$

- propagation over a drift space with gentle (de)magnification
- before slits, ideal lenses and smooth phase elements
- preserves number of pixel and ranges
- given proper sampling, can be used for focusing
- works for strongly astigmatic systems

SRW Quadratic Term/Quadratic special Propagators

Is the Fresnel propagator with analytical treatment of the quadratic phase terms. Uses 2 FFT's

$$u_\omega(x,y,z=z^*) = -\frac{ike^{ikz^*}}{2\pi z^*}\iint_{-\infty}^{+\infty} F(x',y',z=0)e^{\frac{ik}{2}\left[\frac{(x'-x_0)^2}{R_x}+\frac{(y'-y_0)^2}{R_y}+\frac{(x-x')^2+(y-y')^2}{z^*}\right]}dx'dy'$$

Quadratic Term

- propagation over a drift space in general
- before complex optical elements (e.g. curved mirrors).
- preserves number of pixel, ranges are recalculated to accommodate the wavefront
- can be used for or from focusing
- works for strongly astigmatic systems.
-

Quadratic Term Special

- different calculation of Rx and Ry and processing near the waist
- propagation over a drift-spaces in general
- especially adequate when (strongly astigmatic) wavefront is being focused or emerging from very small slits
- strong diffracting elements (gratings)
- preserves number of pixel, ranges are recalculated to accommodate the wavefront
- can be used for or from focusing
- works for strongly astigmatic systems.

SRW From/to Waist Propagators

Is the Fraunhofer Propagator:

$$u_\omega(x,y,z=z^*) = -\frac{ike^{ikz^*}}{2\pi z^*} e^{\frac{ik}{2z^*}(x^2+y^2)} \iint_{-\infty}^{+\infty} u_\omega(x',y',z=0) e^{-\frac{ik}{z^*}(xx'+yy')^2} dx'dy'$$

From Waist

- propagation of a wavefront emerging from a focal position in both vertical and horizontal directions
- output plane (several times) larger than the input plane
- preserves number of pixel, ranges are recalculated to accommodate the wavefront
- fails for strongly astigmatic systems.

To Waist

- propagation of a wavefront being focused on both directions
- output plane (several times) smaller than the input plane
- preserves number of pixel, ranges are recalculated to accommodate the wavefront
- fails for strongly astigmatic systems.

Rules of thumb

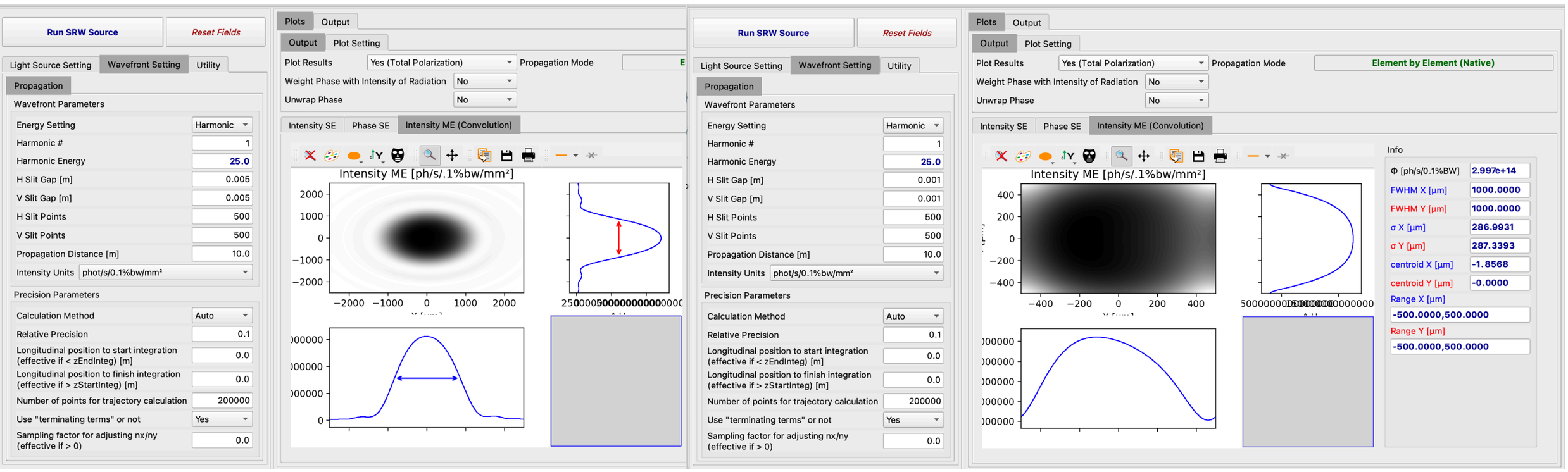
1) Care with the source

2) <u>SEPARATE</u> drift spaces and Optical Elements

3) “Zero’s” padding around OEs

4) DON’T run large ME sims on your computer

1) Care with the source



Choose carefully:

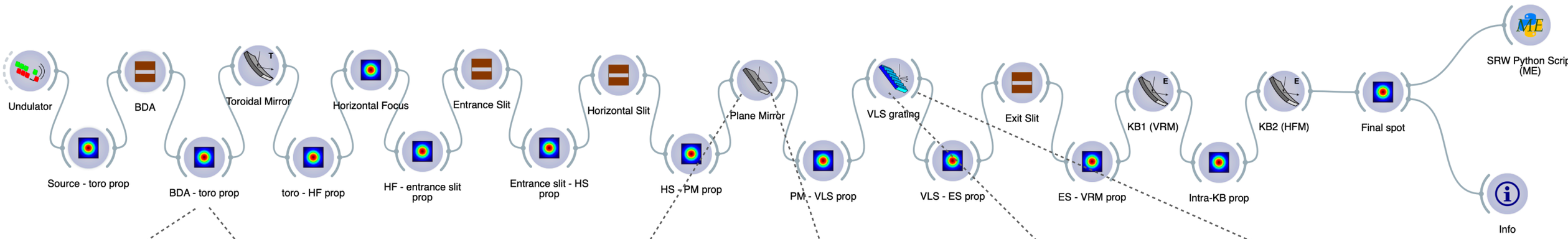
- Size of slit
- Number of points
- Propagation distance
- Pts for traj calculation
- Calc method

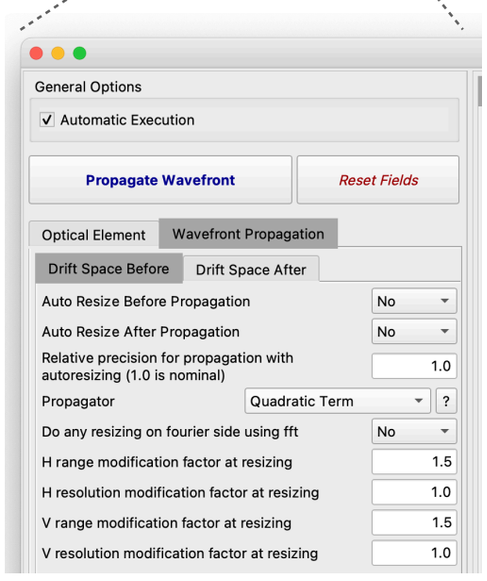
Other things to be aware of

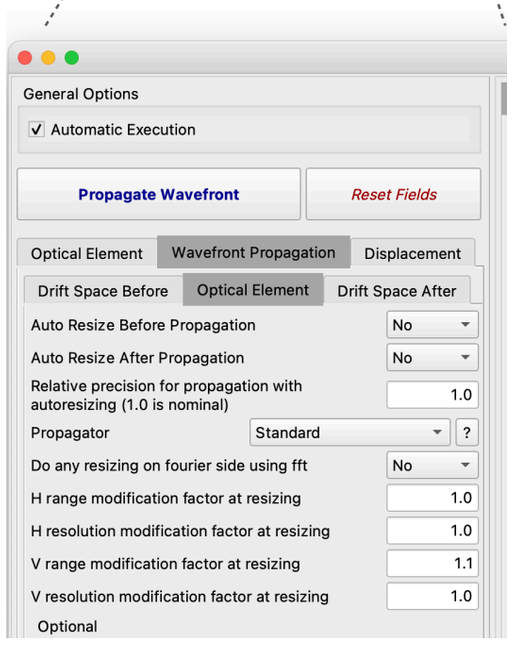
- ID parameters
- Ring and e-beam parameters
- Utility tab (for undulators only).

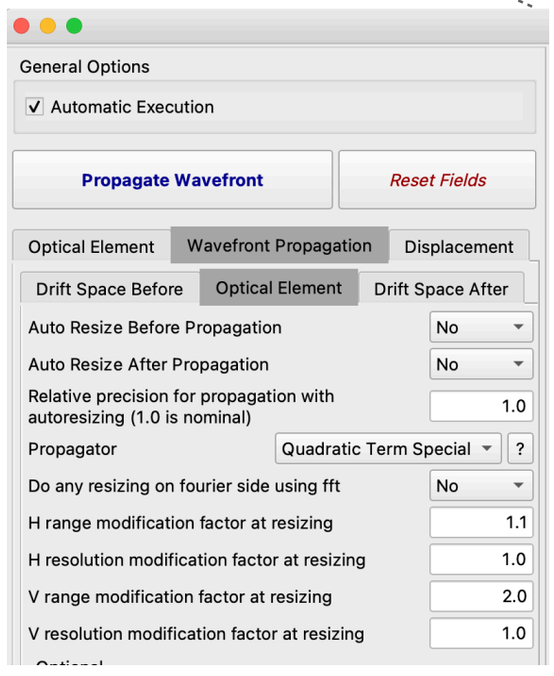
ME (Convolution) doesn’t really mean much…

2) <u>SEPARATE</u> drift spaces and Optical Elements





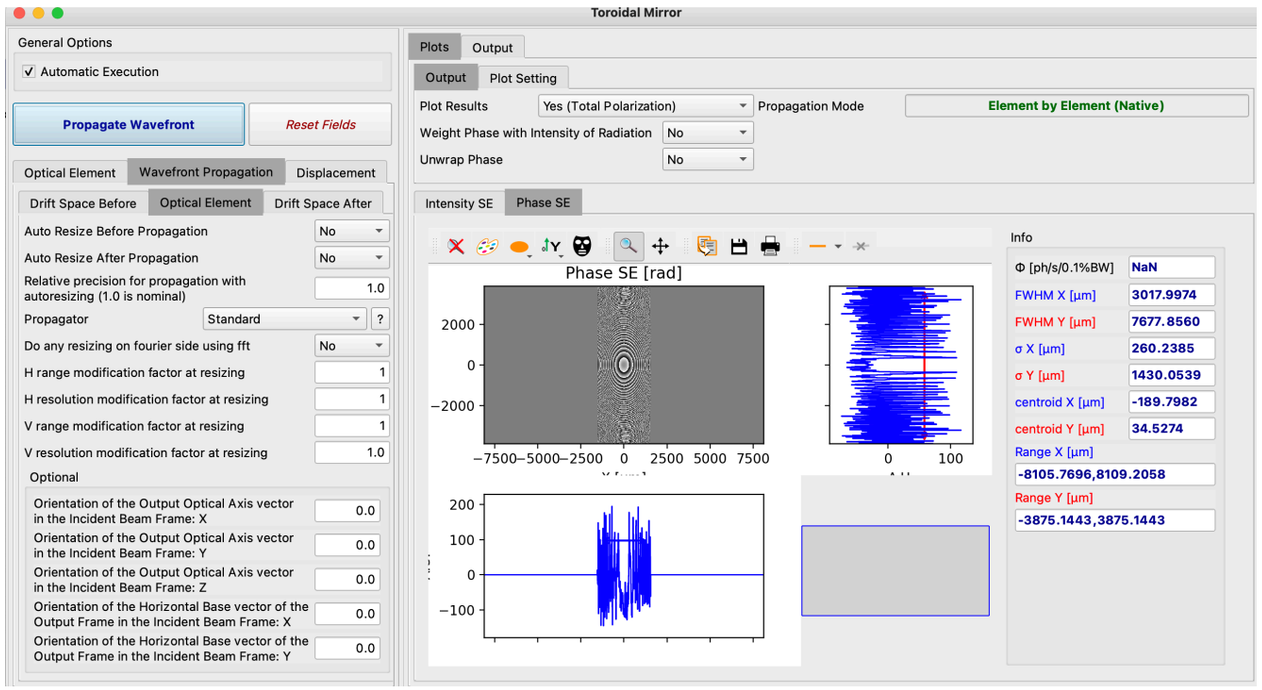


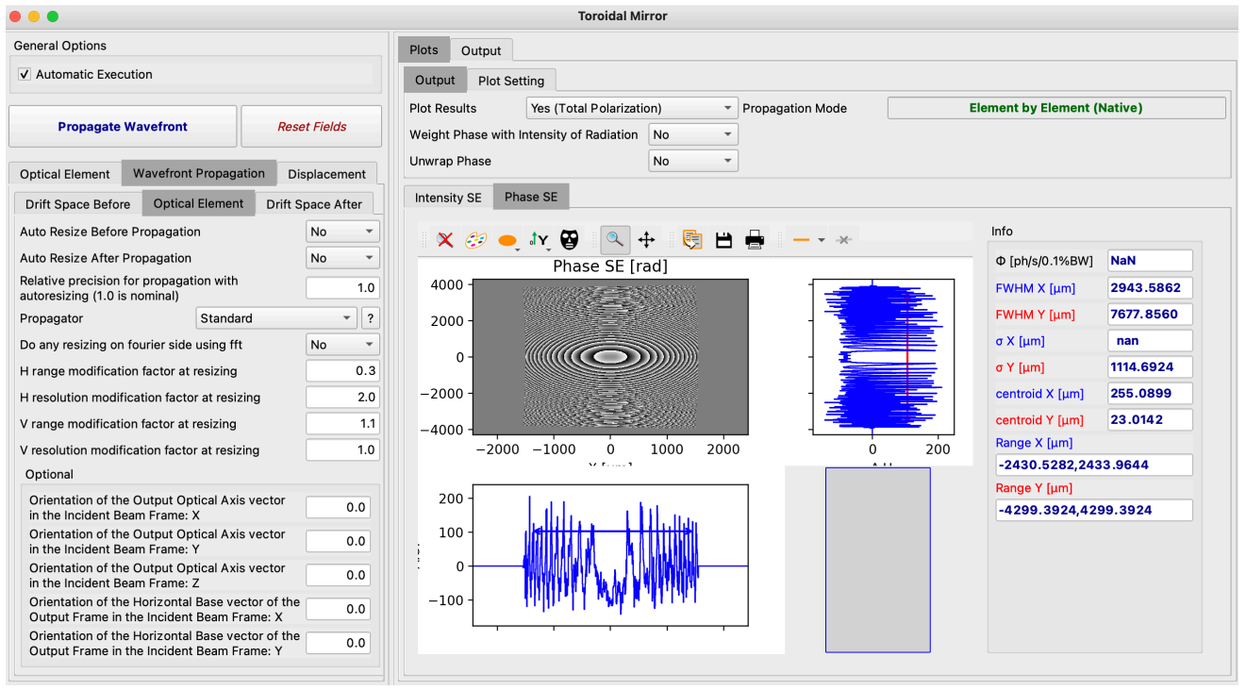


1) Use “Quadratic Term” for drift spaces

2) Use “Standard” propagator for OE’s (except Gratings)

3) “Zero’s” padding around OEs

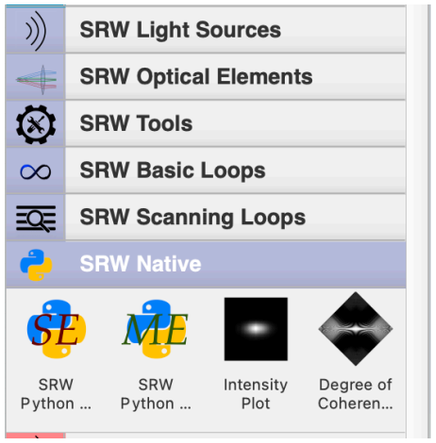


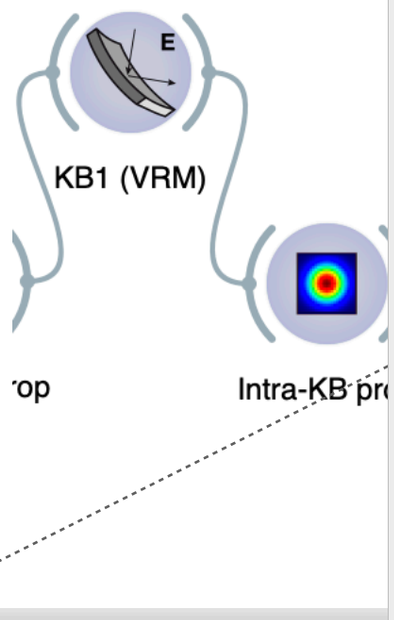


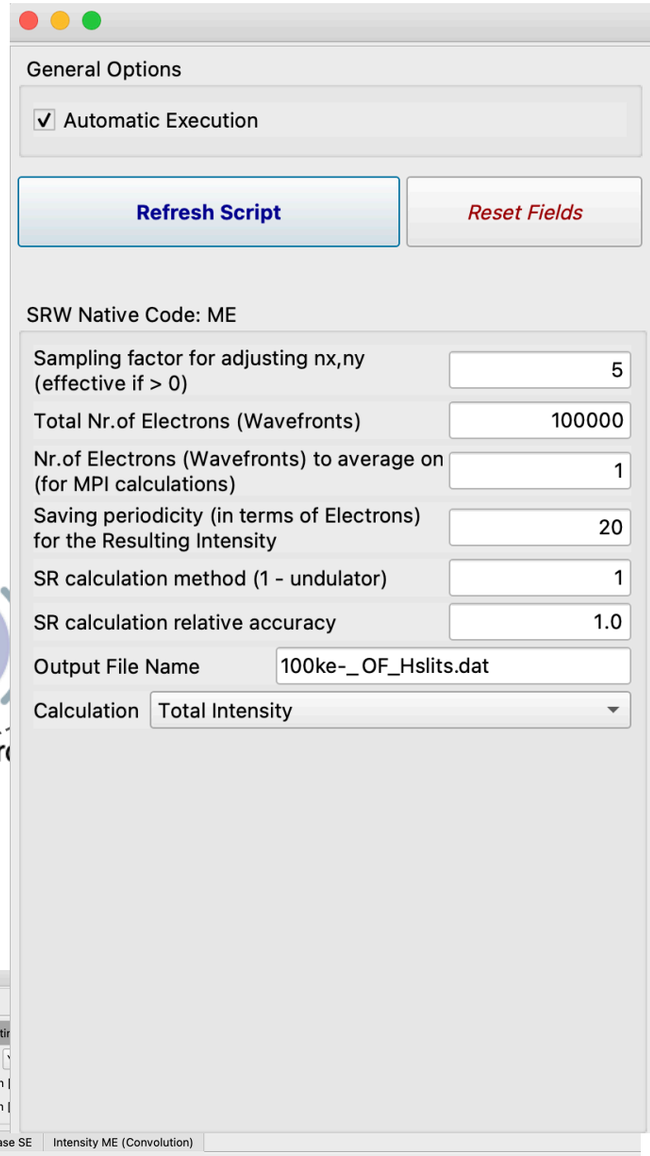
FFT needs a bit of zeroes around.
Too much adds unnecessary computation time!

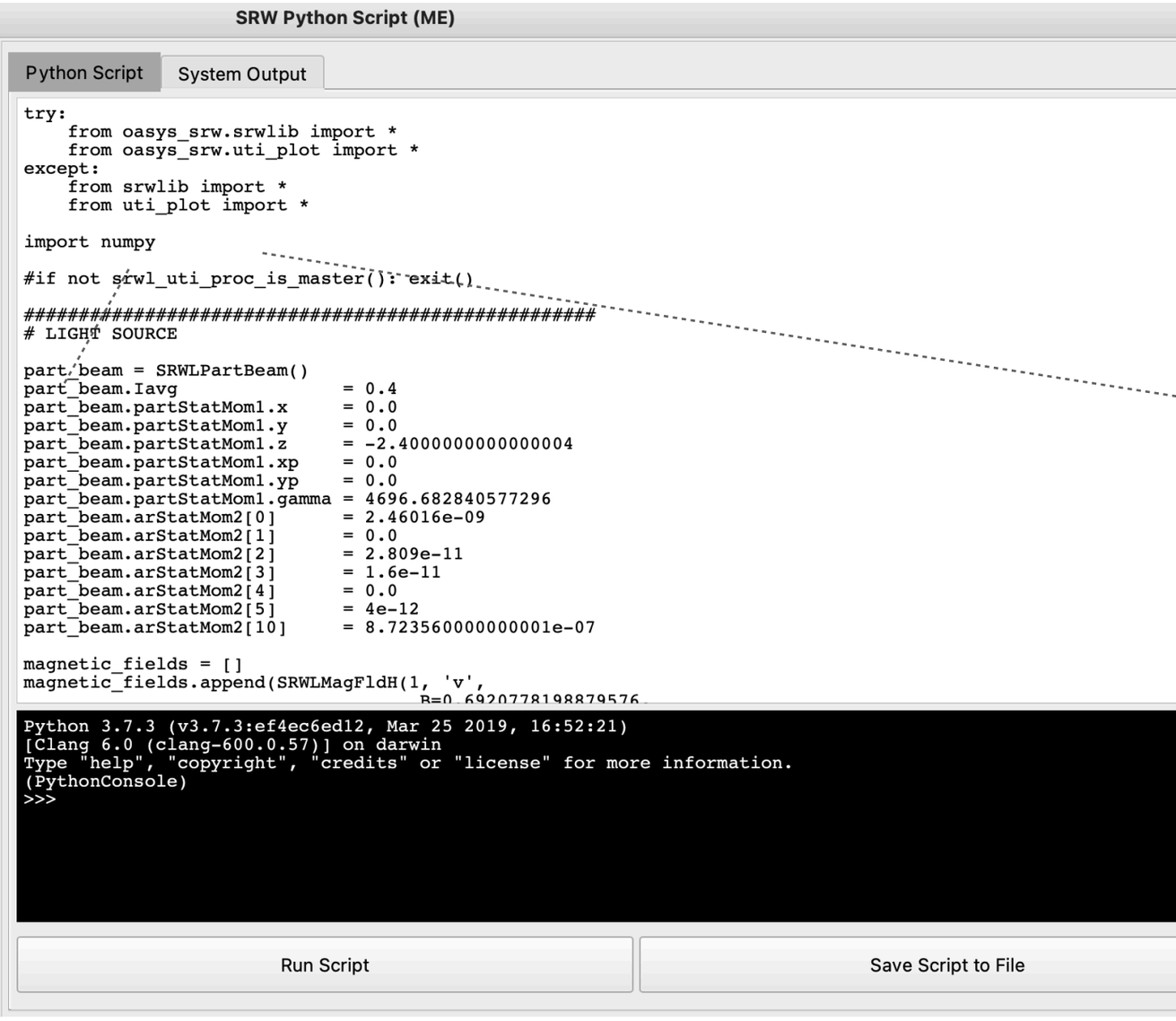
It’s good to play with Range and Resolution

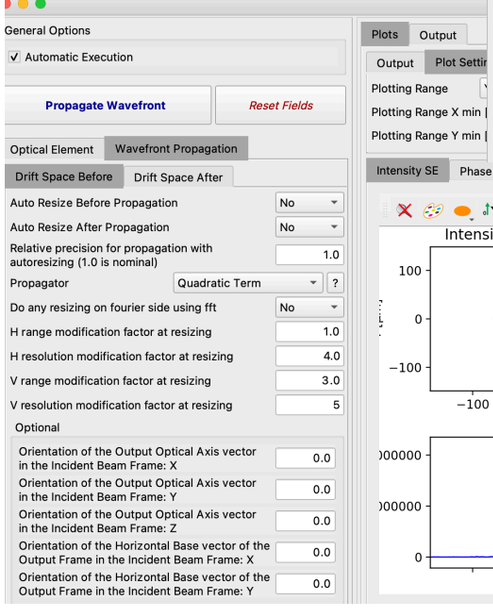
4) DON'T run large ME sims on your computer

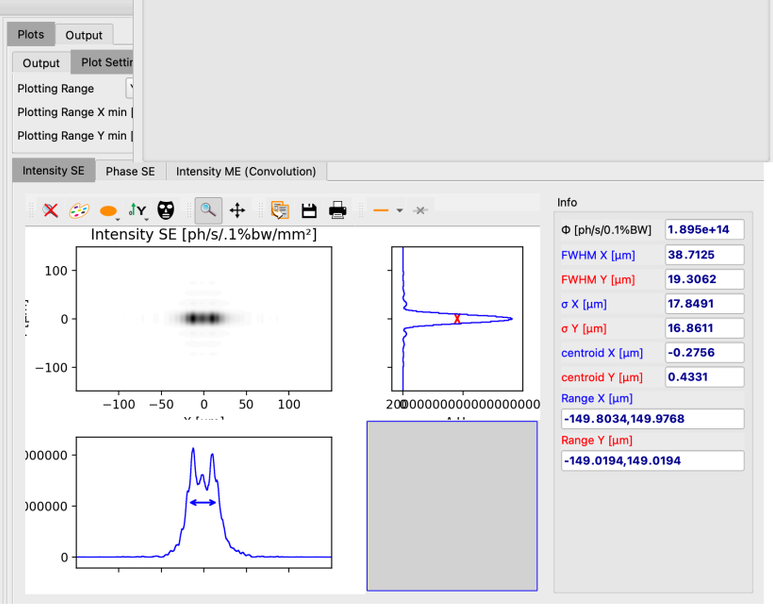












How to run SRW from a terminal

1) **Install SRW via the Oasys distro (from: https://pypi.org/project/OASYS1-SRW/, pip install OASYS1-SRW). SRW is nearly non-compilable on itself.**
2) **Produce either a SE or an ME script from the Oasys interface**

```
try:
    from oasys_srw.srwlib import *
    from oasys_srw.uti_plot import *
except:
    from srwlib import *
    from uti_plot import *

import numpy

#if not srwl_uti_proc_is_master(): exit()
…
```

3) **If SE, you can run direct from the OASYS interface, or through command line**

```
python myOasysScript.py
```

4) **If ME on your machine, you NEED to have OpenMPI and run mpiexec**
5) **Example: 2020 iMac, 1000 electrons ~ 0.7 s/electron**

```
mpiexec -n 9 python3.8 test3.py &

cat __srwl_logs__/*.log

[2023-01-16 13:26:25]: Calculation on 9 cores with averaging of 1 particles/iteration.
[2023-01-16 13:26:34]: Running     1 out of 1000 (  0.10% complete)
…

[2023-01-16 13:38:31]: Finished 1000 out of 1000 (100.00% complete)n
```

6) **For large computations (i.e. lots of e-) must go on kalculus**

You can write your own scripts, but not recommended for full beamlines

A couple of words on kalculus

Elettra's in-house cluster:

- 400 core (800 thread) CPU / 6.5 TB RAM on 8 blades;
- 13824 CUDA core, 864 Tensor core and 80GB RAM GPU HBM2 on 2 blades;

Need to have:
a. runnable .py script (check on your machine with ~1/5 electrons first),
b. .submit file (more in a sec)
c. Elettra's email credentials
d. plugged network connection from inside Elettra OR
e. a VPN if connected via wifi
f. all the files needed inside the directory on the cluster

1) log in via SSH with your Elettra's email credentials:

ssh -X name.surname@kalculus

2) Check status of the cluster

```
[matteo.altissimo@k-submission NanoSpec2023]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
integral     up 3-00:00:00      1  down* k-bc01-nc17
integral     up 3-00:00:00      1   comp k-bc01-nc06
integral     up 3-00:00:00      1    mix k-bc01-nc04
integral     up 3-00:00:00     14  alloc k-bc01-nc[01-03,05,07-16]
```

3) Start your job

sbatch yourfile.submit

.submit file

```
#!/bin/bash
#SBATCH --job-name=100ke-_OF_Hslits _________
#SBATCH --output=100ke-_OF_Hslits ___________
#
#SBATCH --partition=integral ________________
#SBATCH --time=72:00:00 _____________________
#SBATCH --nodelist=k-bc01-nc[05,07-09,15-16] _
#SBATCH --mincpus=96 ________________________

module load software/SRW-git-python3.6 _____
srun  python3 100ke-_OF_HSlits.py ___________
```

Name of job

File for errors

Partition name

Time limit

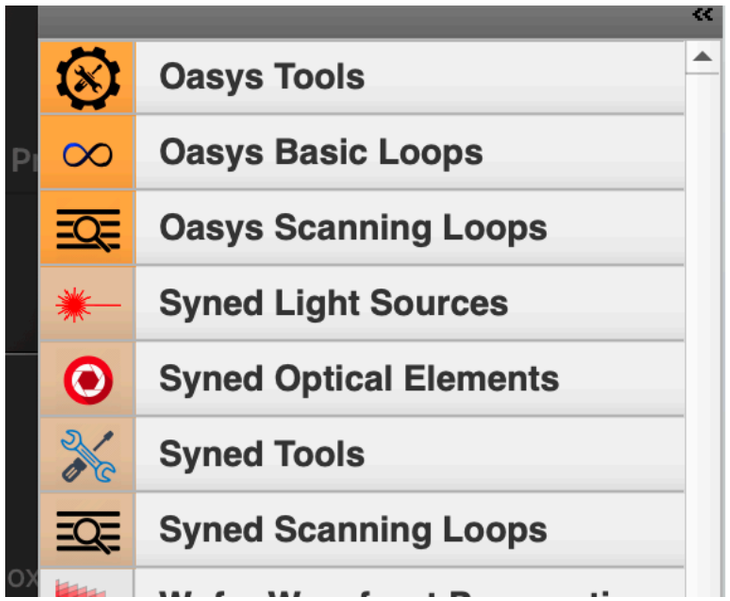
Nodes's list: [i-j] or [i,j,z,q]

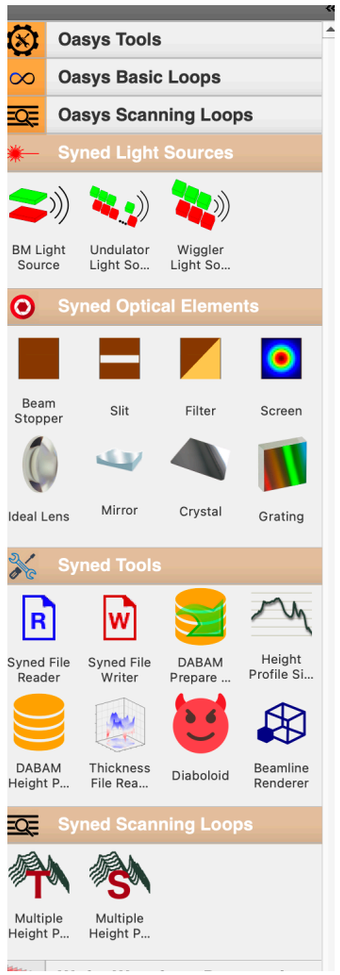
# of cpus used per node

Pre-loading of SRW

Running your script

SYNED: step towards standardisation

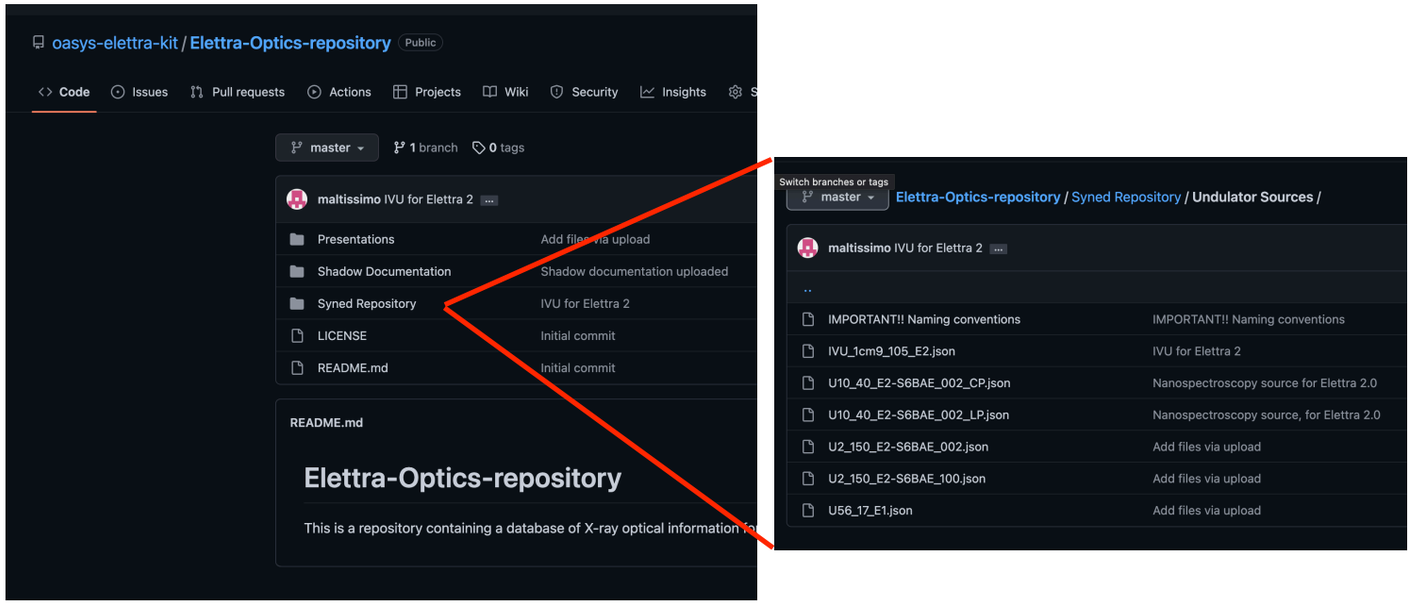




- Uses a common language (i.e. a .json file) to apply the properties of an abstract object to a corresponding SHADOW/SRW/WOFY/WISEr
- Can store on the cloud

A number of sources are available for Elettra 2.0 on Github’s Elettra Optics repo

https://github.com/oasys-elettra-kit/Elettra-Optics-repository



Naming convention

TypePeriod(cm)_#OfPeriods_StorageRing_Lattice_coupling.json.

|_____| |_____|

ID Storage Ring

An example:

U2_150_E2-S6BAE_002.json

U = Undulator

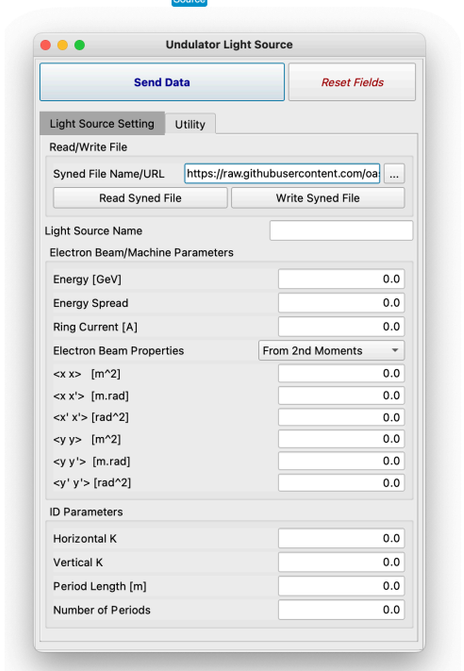
2= 2 cm period

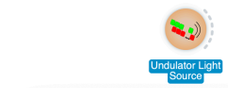
150= 150 periods

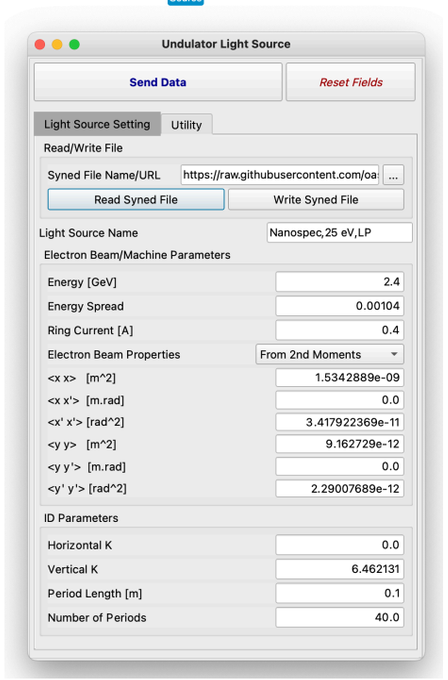
E2= Elettra 2

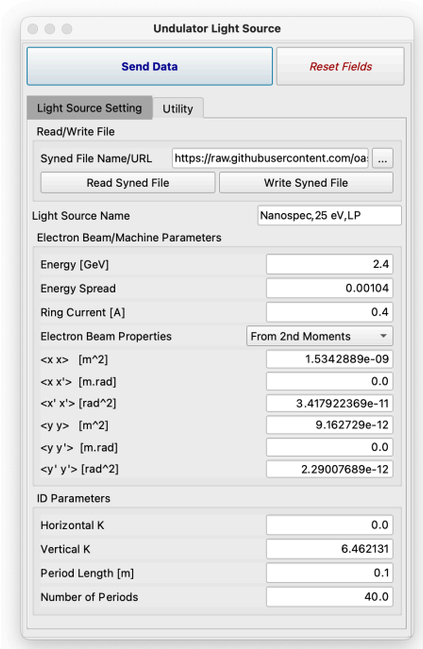
S6BAE= Special 6 Bends Achromat Enhanced

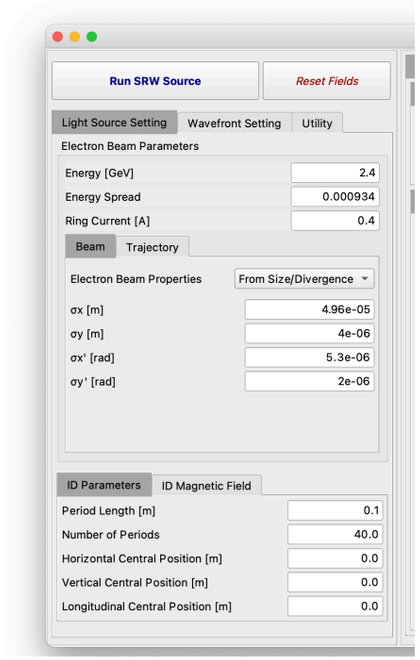
002=2% coupling











SRW propagators

Assume that the electric field before the propagation has quadratic terms in its phase, defined by the radii of the wavefront curvature in the horizontal and vertical planes R_x, R_y and the transverse ordinates of the center point (x_0 ,y_0)

$$u_\omega(x,y,z=0) = F(x,y,z=0)e^{\frac{ik}{2}\left[\frac{(x-x_0)^2}{R_x}+\frac{(y-y_0)^2}{R_y}\right]}$$

The propagated wavefield becomes:

$$u_\omega(x,y,z=z^*) = -\frac{ike^{ikz^*}}{2\pi z^*}\iint_{-\infty}^{+\infty} F(x',y',z=0)e^{\frac{ik}{2}\left[\frac{(x'-x_0)^2}{R_x}+\frac{(y'-y_0)^2}{R_y}+\frac{(x-x')^2+(y-y')^2}{z^*}\right]}dx'dy' =$$

$$= F(x,y,z=z^*)e^{\frac{ik}{2}\left[\frac{(x-x_0)^2}{R_x+z^*}+\frac{(y-y_0)^2}{R_y+z^*}\right]}$$

Where:

$$F(x,y,z=z^*) = -\frac{ike^{ikz^*}}{2\pi z^*}\iint_{-\infty}^{+\infty} F(x',y',z=0)e^{\frac{ik}{2z^*}\left[\frac{R_x+z^*}{R_x}\left(\frac{R_x x+z^* x_0}{R_x+z^*}-x'\right)^2+\frac{R_y+z^*}{R_y}\left(\frac{R_y y+z^* y_0}{R_y+z^*}-y'\right)^2\right]}dx'dy'$$

Chubar, O. and Celestre, R., Opt. Express 27, 28750-28759 (2019)