panosc
photon and neutron
open science cloud

Development of WISEr wave propagation code within OASYS

7th November, 2019

Author: Aljoša Hafner, CERIC-ERIC

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

Outline

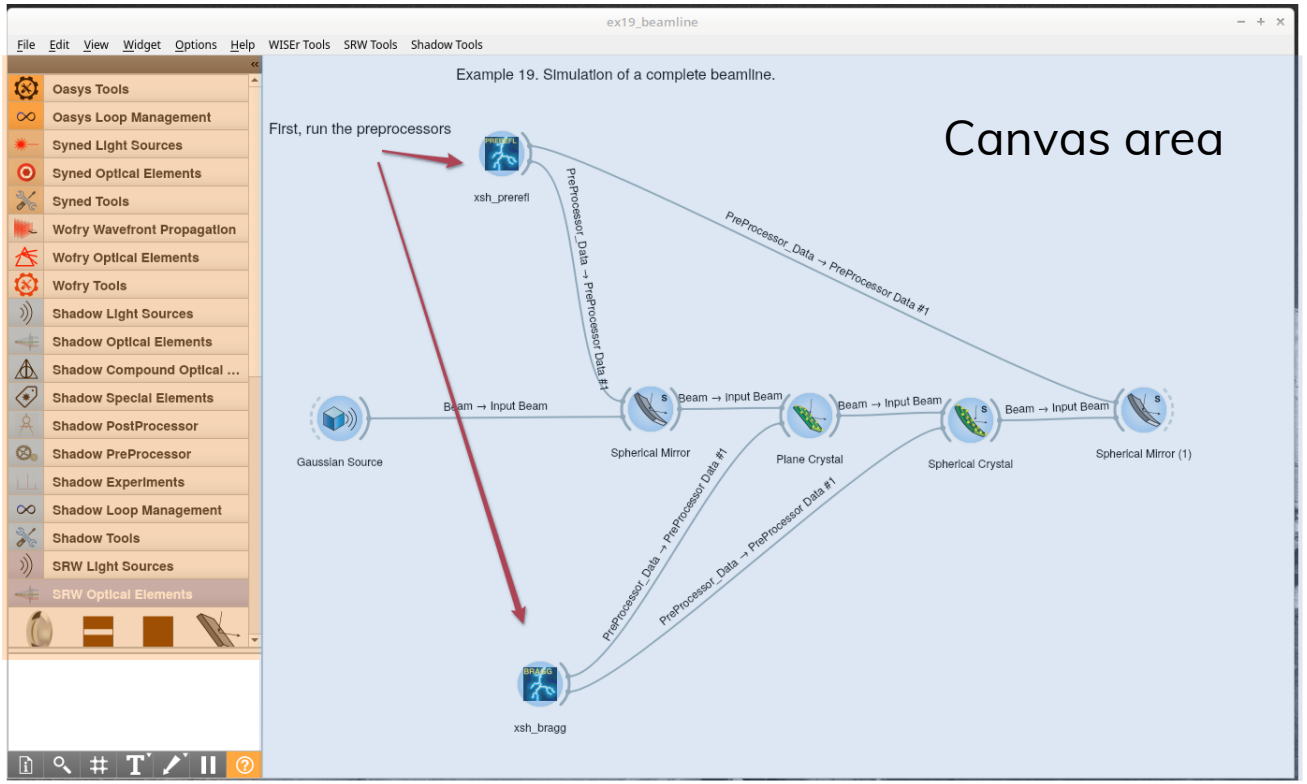
- **OASYS** demonstration and overview
- **WISEr** code for numerical integration of wave propagation
- Implementation of numerical optimization package – **numba**

panosc

OASYS Demonstration

- Quick **OASYS** demonstration and overview
  - Framework for Orange widgets for synchrotron optics simulations
  - Extend with widgets (stored on PyPi)
  - Libraries: Python, Qt
    - Also external libraries: SRW (wave propagation with FT)

Widgets



WISEr code presentation

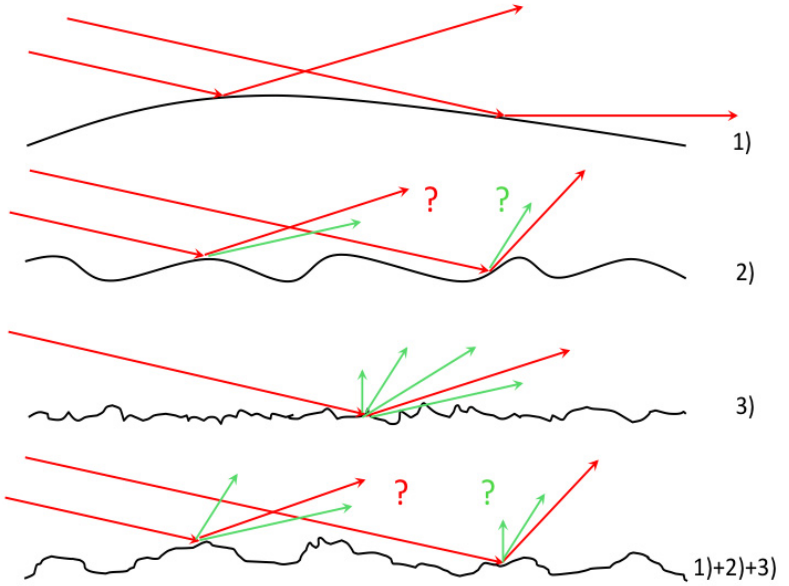
- Numerical integration, based on Fresnel-Huygens principle
  - Raimondi, L., Spiga, D. A&A 573 (2015)
- Grazing incidence optics
- (Partial) coherence

$$E(x,0,z) = \frac{E_0\,\Delta R_1}{L_1\sqrt{\lambda x}}\int_f^{f+L_1}\sqrt{\frac{x_1}{\bar{d}_2}}\,\mathrm{e}^{-\frac{2\pi i}{\lambda}\left[\bar{d}_2 - z_1 + \frac{x_1^2}{2(S-z_1)}\right]}\,\mathrm{d}z_1, \qquad (6)$$

where ΔR_1 is given by Eq. (3), we have omitted unessential phase factors, evaluated the radial coordinate at $x_1(z_1)$, and defined

$$\bar{d}_2 = \sqrt{(x_1 - x)^2 + (z_1 - z)^2}. \qquad (7)$$

- No contributions anymore from original authors
- Restarted activity on github and development

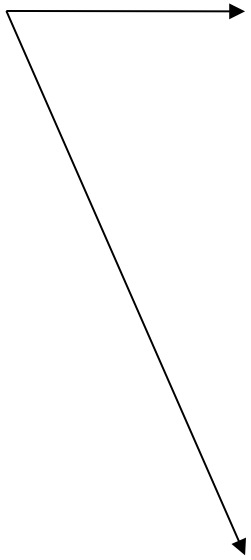


1) Figure error – geometrical optics
2) ??
3) Roughness – first-order scattering theory
1) + 2) + 3) ????

panosc

Numba – easy way to parallelize and use GPUs

- Decorators: jit (just-in-time)
- Implementation of numerical optimization package – **numba**
  - Easy and straightforward speed-up of the code
  - Multi-core and GPU calculations



```
kT = 2 / math.log(1 + math.sqrt(2), math.e)

@numba.jit(nopython=True)
def update_one_element(x, i, j):
    n, m = x.shape
    assert n > 0
    assert m > 0
    dE = 2 * x[i, j] * (
                      x[(i-1)%n, (j-1)%m]
                    + x[(i-1)%n,  j     ]
                    + x[(i-1)%n, (j+1)%m]

                    + x[ i     , (j-1)%m]
                    + x[ i     , (j+1)%m]

                    + x[(i+1)%n, (j-1)%m]
                    + x[(i+1)%n,  j     ]
                    + x[(i+1)%n, (j+1)%m]
                    )
    if dE <= 0 or exp(-dE / kT) > np.random.random():
        x[i, j] = -x[i, j]

@numba.jit(nopython=True)
def update_one_frame(x):
    n, m = x.shape
    for i in range(n):
        for j in range(0, m, 2):  # Even columns first to avoid overlap
```

Thank you

Aljoša Hafner
CERIC-ERIC
aljosa.hafner@ceric-eric.eu

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852