Documentation for 2D X-ray Beamline package (xraybeamline2d)

Matt Seaberg

April 9, 2020

1 Introduction

The software described here is inspired by SRW in that it uses the Fresnel scaling theorem to propagate a 2D wavefront between optical devices that are generally described as phase/amplitude screens. This is typically a good approximation in the case of X-ray wavelengths, that are slowly diverging. The difference with SRW is that the software described here is written entirely in Python, which may have both advantages and disadvantages, and that it has been developed to be easy to simulate misalignments with similar inputs to what is used in Shadow. Where possible, I will make comparisons with Shadow to validate output of xraybeamline2d. Additionally, the necessary devices to simulate the new L2SI beamlines have been prioritized.

2 beam module

2.1 Beam class

2.1.1 Initialization and attributes

The Beam class stores all of the wavefront information as it is propagated through the various optics/devices along a beamline. The horizontal and vertical propagation angles (linear phase) and horizontal and vertical distance to focus (quadratic phase) are tracked separately from the high order phase, allowing the propagation of 2D planes with modest sampling requirements, as is the case with SRW. A Beam is initialized with an optional 2D complex numpy array (input), and a beam parameter dictionary (beam_param). All quantities are in SI units unless otherwise specified. See Table 1 for relevant dictionary entries when "input" is provided. Keys are required unless specified as optional.

In the case that "input" is not provided, the dictionary entries shown in Table 2 are needed, in addition to those shown in Table 1. In this case, "dx" (horizontal pixel size)

Table 1: Entries for beam_param dictionary when "input" is provided.

Key	**Description**
cx	horizontal beam center at initial plane (optional, default 0)
cy	vertical beam center at initial plane (optional, default 0)
ax	horizontal beam propagation angle (optional, default 0)
ay	vertical beam propagation angle (optional, default 0)
dx	horizontal pixel size
dy	vertical pixel size (optional, defaults to horizontal pixel size)
photonEnergy	beam photon energy (eV)
rangeFactor	factor controlling where at which point to switch between propagation via Fresnel scaling, versus typical unscaled propagation. This is in units of Rayleigh lengths from the focus. (optional, default 10)

Table 2: Additional entries for beam_param dictionary when "input" is not provided.

Key	**Description**
N	grid size in pixels (over-ridden if complex input array is provided)
sigma_x	beam width (1/e radius in field strength) at waist
sigma_y	beam width (1/e radius in field strength) at waist
z0x	distance from horizontal waist at initial plane
z0y	distance from vertical waist at initial plane

is optional. If not provided, the field of view is set to be eight times wider than the full width at half maximum in the initial plane.

In the case of beam propagation through a beamline, the typical mode of operation is to initialize a Beam object, which is then passed to a Beamline object (see Section 4.1), which handles the propagation through the various devices along the beamline. However, there may be cases where direct interfacing with a Beam object is useful. See Table 3 for a list of Beam object attributes.

2.1.2 Available methods

The Beam class has several methods; most of them are only relevant for internal use. The "beam_prop" method is heavily used for propagating beams through a Beamline (Section 4.1), and can be used directly as well. The syntax is:

```
b1.beam_prop(dz),
```

where b1 is a Beam object and dz is the distance to propagate in meters. The other methods that may be used externally are rescale_x and rescale_y. These are relevant in the case of gratings or miscut crystals, and are used as

```
b1.rescale_x(factor),
```

where the beam is scaled to be larger by an amount factor. In this case the beam center is also scaled, as would be the case in diffraction from a grating.

2.1.3 Propagation method

As mentioned above, the type of propagation used for the Beam class is based on the Fresnel scaling theorem, which is valid in cases where the paraxial approximation applies. This states that for a diverging (or converging) beam can be propagated as a plane wave (factoring out the quadratic term of the wavefront), in the following way. We discuss the 1D case, the 2D case is a straightforward extension. We introduce a magnification factor:

$$M = \frac{R_x + z}{R_x}, \tag{1}$$

where R_x is the beam's horizontal radius of curvature, and z is the distance to propagate. The plane wave is propagated by a distance

$$z_{Eff} = z/M \tag{2}$$

and in the resulting plane the x coordinates are scaled by

$$x_f = x_i M, \tag{3}$$

Table 3: Beam object attributes.

Attribute	**Description**
wave	a 2D, complex-valued numpy array with amplitude and phase information. Outside the focus region, this only includes high-order phase information. Inside the focus, it includes quadratic phase and higher.
x	horizontal coordinates (2D numpy array with same shape as wave)
y	vertical coordinates (2D numpy array with same shape as wave)
zx	horizontal wavefront radius of curvature
zy	vertical wavefront radius of curvature
cx	horizontal beam center
cy	vertical beam center
ax	horizontal propagation angle
ay	vertical propagation angle
focused_x	boolean state, true if Beam is currently inside horizontal focus region, false if Beam is currently outside horizontal focus region
focused_y	boolean state, true if Beam is currently inside vertical focus region, false if Beam is currently outside vertical focus region
photonEnergy	photon energy (eV)
lambda0	wavelength
k0	wavenumber ($2\pi/\lambda$)
zRx	horizontal Rayleigh range multiplied by rangeFactor, based on current divergence.
zRy	vertical Rayleigh range multiplied by rangeFactor, based on current divergence.
rangeFactor	factor controlling at which distance from the focus to begin normal propagation

where x_f corresponds to the x coordinates in the new plane after propagation, and x_i corresponds to the x coordinates in the initial plane. The beam's radius of curvature simply becomes

$$R_{xf} = R_{xi} + z. \tag{4}$$

As is clear from Eq. 1, near the focus it is possible to have singularities in Eqs. 2 and 3. To avoid this situation, the approach we take with the Beam class is to propagate "normally" at planes that are within some factor of the Rayleigh range from the focus. This means that if the beam starts out unfocused and the goal is to propagate within this range, the beam is first propagated to the boundary defined by the (adjustable) multiple of the Rayleigh range via the Fresnel scaling theorem, and then propagated normally the rest of the way. Conversely, if the beam starts out inside this focus range, and the goal is to propagate to some plane outside the range, it is first propagated normally to the boundary of the range, and then propagated the rest of the way using the scaling method. The main complication with the extension to 2D is that the propagation may need to be split into more than 2 steps in cases with astigmatism.

2.2 GaussianSource class

The GaussianSource class is a helper class used for initializing a Beam object. This object is created in the initialization step of the Beam class. To initialize a GaussianSource object, a single beam_param argument is required, with the same entries as are given in Tables 1 and 2. The main thing to consider here is that, without initial input to a Beam object, the GaussianSource class sets the field of view of the beam. If the pixel size (dx) is not defined in the beam_param dictionary, the field of view is set to be 8 times the full width at half maximum (FWHM) of the beam at the initial plane. In the case where the beam is initialized within the focus region, the field of view is set such that it will be 8 times the FWHM when the beam is propagated normally to the boundary of the focus region. If the pixel size is defined, this behavior is overridden.

3 optics module

The optics module is meant for use with the Beam class, in the sense that a Beam object can be influenced by, or "propagated through", a given optical element. Each type of optical element or device has an associated class. Each class is required to have a "propagate" method, which takes a Beam object as its only argument. Another commonality to all devices is that they must all have a name attribute and a "z" attribute (corresponding to the z location along the beamline). To initialize any optical element, the name (a string) is given as the first argument, and any additional arguments are given as keyword arguments.

3.1 Mirror class

The Mirror class is a parent class for FlatMirror (Section 3.2) and CurvedMirror (section 3.3), and includes attributes and methods that are common to all mirror types. Mirrors are currently assumed to be positioned at glancing incidence, currently with large glancing angles (where the small angle approximation doesn't apply) not guaranteed to be supported. See Fig. 1 for the relationship between mirror and beam coordinates.

See Fig. 2 for reference on the effect of mirror translation relative to its normal vector. There are two main effects: the beam is centered on a different point on the mirror, and the beam is translated upon reflection in the beam's x-axis direction. As can be seen from the figure, a mirror motion δx_m causes the beam to be shifted (in mirror coordinates) along the mirror's z-axis by

$$\delta z_m = -\frac{\delta x_m}{\tan\alpha}. \tag{5}$$

Secondly, the beam is translated by

$$\delta x_b = 2\delta x_m \cos\alpha. \tag{6}$$

There is also a minor effect of a change in the beam's path length. The change is simply given by $\overline{AD} - (\overline{AB} + \overline{BC})$:

$$\delta z_b = \frac{2\delta x_m}{\sin\alpha}\cos^2\alpha - \frac{2\delta x_m}{\sin\alpha} = -2\delta x_m \sin\alpha. \tag{7}$$

For small adjustments δx_m and small glancing angles this is smaller than a typical Rayleigh range, so it is safe to ignore this small change in path length.

Another consideration is the effect of a change in the beam's center relative to the beam x-axis. See Fig. 3 for reference. From the figure it is clear that for a shift in the beam's center, δx_b, the beam moves by a distance δz_m on the mirror surface given by

$$\delta z_m = \frac{\delta x_b}{\sin\alpha}. \tag{8}$$

Secondly, upon reflection the beam center on the beam's x-axis changes sign, as depicted in Fig. 3.

In order to account for phase errors in the reflected beam due to height errors on a surface, the change in path length due to the height error must be calculated. See Fig. 4 for reference. In the figure, α is the angle of incidence. For convenience the angle γ is defined as

$$\gamma = \frac{\pi}{2} - 2\alpha \tag{9}$$

As mentioned in the caption for Fig. 4, the path length difference is given by the difference in segment lengths $\overline{AB} - \overline{AC}$. The segment lengths are given by:

$$\overline{AC} = \frac{h}{\sin\alpha} \tag{10}$$

$$\overline{AB} = \frac{h}{\sin\alpha}\sin\gamma = \frac{h}{\sin\alpha}\left(1 - 2\sin^2\alpha\right) \tag{11}$$

The path length difference is then given by

$$\Delta z = \overline{AB} - \overline{AC} = -2h\sin\alpha \tag{12}$$

with a resulting phase change of

$$\Delta\phi = -\frac{4\pi}{\lambda}h\sin\alpha. \tag{13}$$

An analogous expression can be found for the effects of a height error on a grating surface. The difference is that the diffracted beam exits at a different angle, leading to slight change in the path length. See Fig. 5 for reference. The path length difference is again $\overline{AB} - \overline{AC}$. Here the angle γ is slightly different. It is now

$$\gamma = \frac{\pi}{2} - \beta - \alpha \tag{14}$$

The length of $\overline{AC}$ is still given by Eqn. 10. The length of $\overline{AB}$ becomes

$$\overline{AB} = \overline{AC}\sin\gamma = \frac{h}{\sin\alpha}\sin\gamma \tag{15}$$

This gives a path length difference of

$$\Delta z = \overline{AB} - \overline{AC} = \frac{h}{\sin\alpha}\left(\sin\gamma - 1\right) = \frac{-2h}{\sin\alpha}, \sin^2\left(\frac{\alpha+\beta}{2}\right) \tag{16}$$

giving a phase change of

$$\Delta\phi = \frac{-4\pi h}{\lambda\sin\alpha}\sin^2\left(\frac{\alpha+\beta}{2}\right) \tag{17}$$

We see that when $\alpha = \beta$ as in the case of a mirror this reduces to the result obtained in Eq. 13.

3.2 FlatMirror class

3.3 CurvedMirror class

3.4 Grating class

3.5 Mono class

3.6 Collimator class

3.7 Slit class

3.8 Drift class

3.9 CRL class

3.10 PPM class

PPM is facing upstream.

4 beamline2d subpackage

4.1 Beamline class

5 Future Developments

Future developments will include

- a Pulse class in order to propagate for example SASE pulses through a beamline, and see the effect on the time structure.
- Methods to interfere two beams/pulses and view the resulting interference pattern.
- Tracking device and beam positions in absolute coordinates, along with some form of viewing this in 3D.
- Crystals for hard X-ray monochromators, including miscut feature.
- Additional mirror types such as ellipsoids.
- Better simulation of elliptical mirror including the unequally spaced mapping of beam coordinates to mirror coordinates.

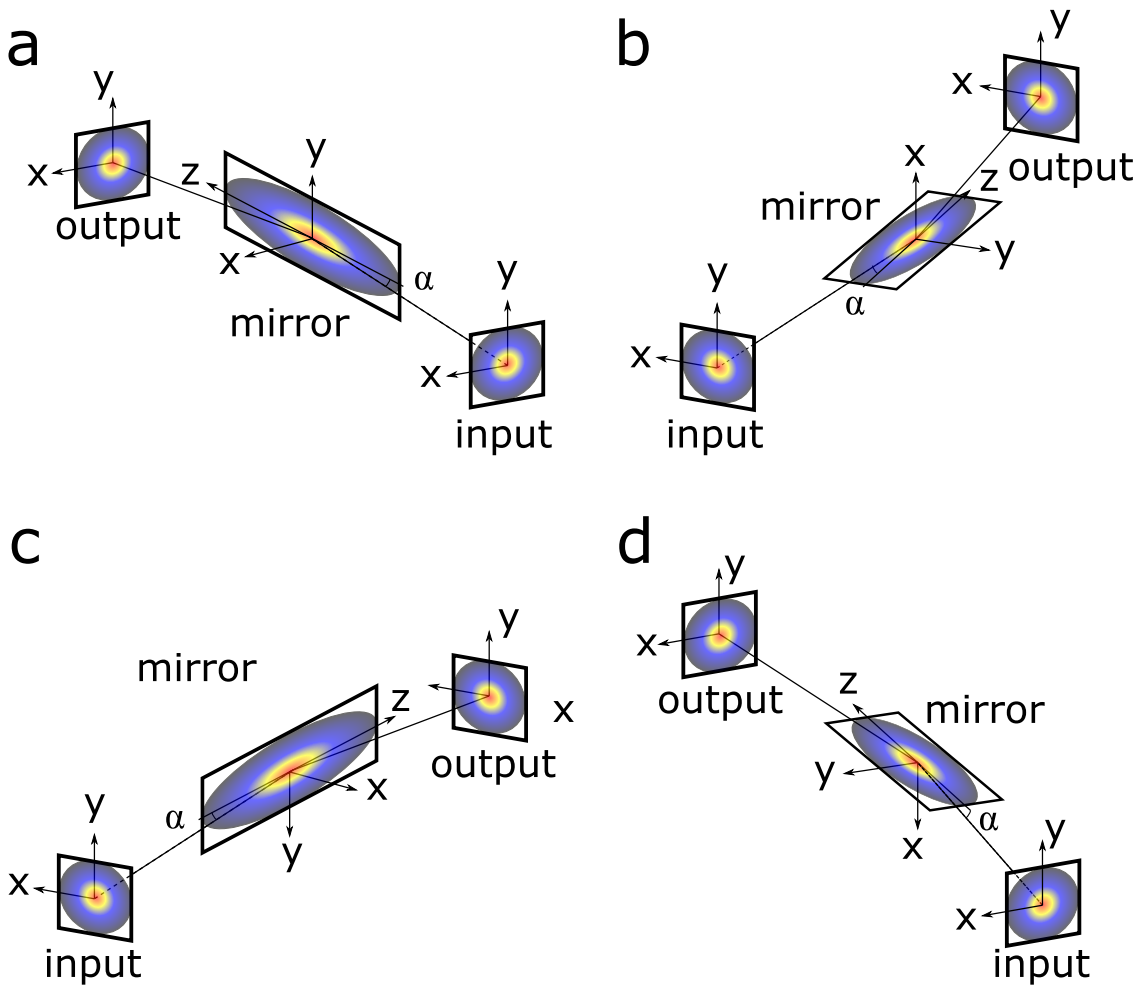

Figure 1: Coordinate system definition for beam and mirror. Beam coordinates follow the LCLS convention, with the z direction pointing in the direction of propagation. For the mirror coordinates, the x-axis is always defined to be the surface normal, and the z-axis is always defined to point in the direction of propagation, roughly parallel to the beam's z-axis. (a) 0° orientation, with the mirror y-axis aligned with the beam's y-axis, with angle of incidence α (referred to as orientation "0"). (b) 90° orientation (y-axis is rotated 90° about the z-axis relative to (a)), with angle of incidence α (orientation "1"). (c) 180° orientation (orientation "2"). (d) 270° orientation (orientation "3"). A positive angle is defined to be a counter-clockwise rotation about a given axis. The orientation in (a) results in a deflection of the beam in the positive x-direction, and (b) results in a deflection of the beam in the positive y-direction. The orientations in (c) and (d) result in deflections in the negative x- and y-directions, respectively.

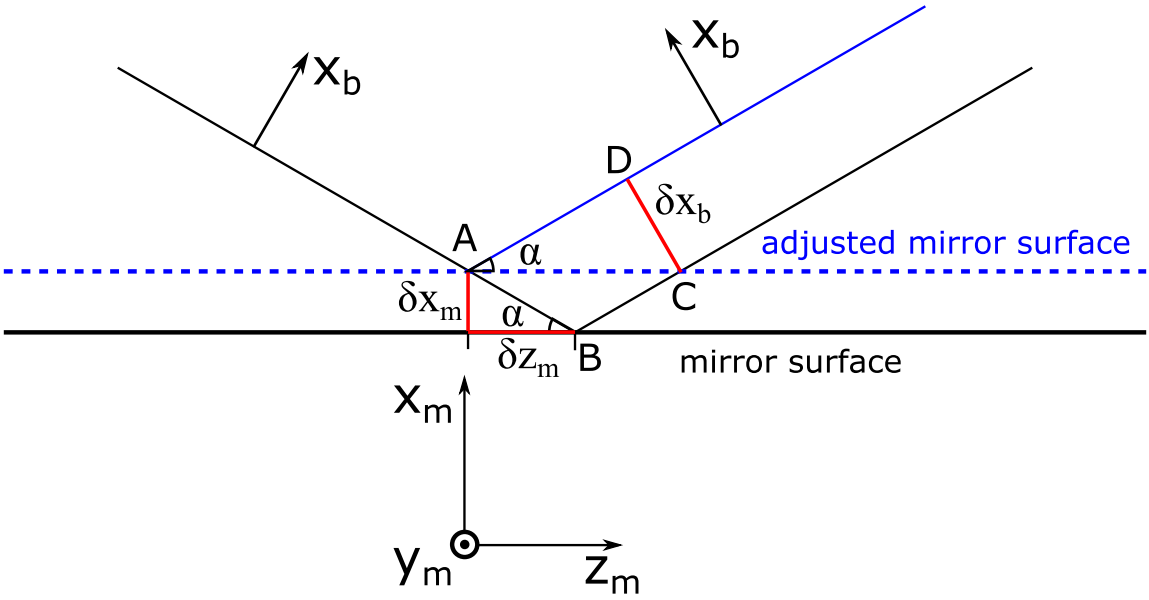

Figure 2: Effect of mirror translation parallel to the surface normal (mirror x-axis). Note the vertices A, B, C, and D that are referenced in the text.

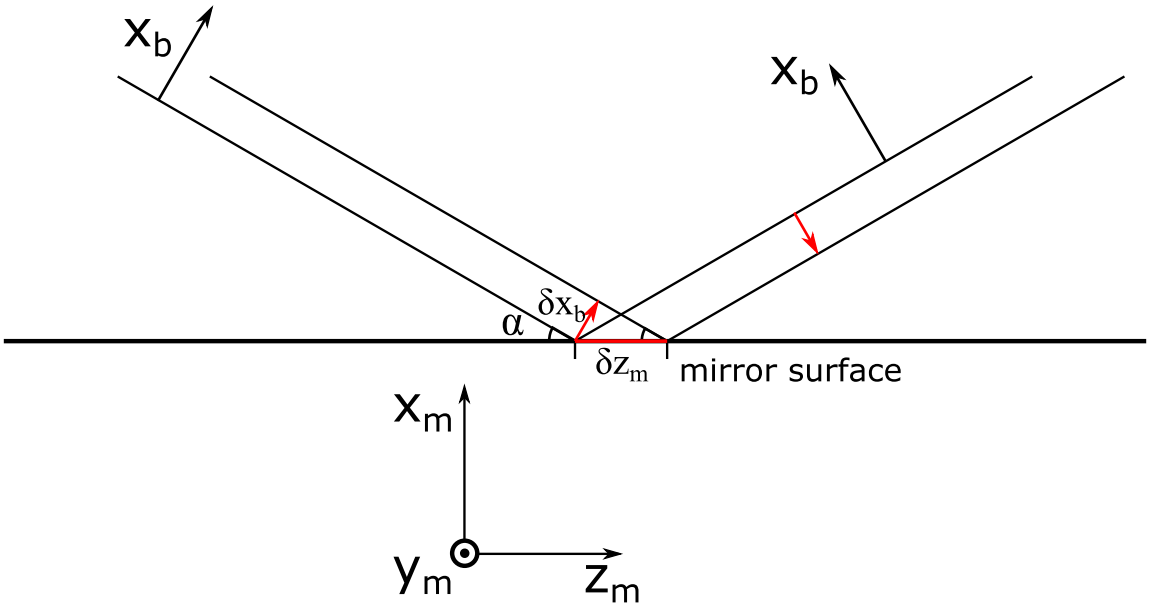

Figure 3: Effect of beam center shift in the plane of incidence.

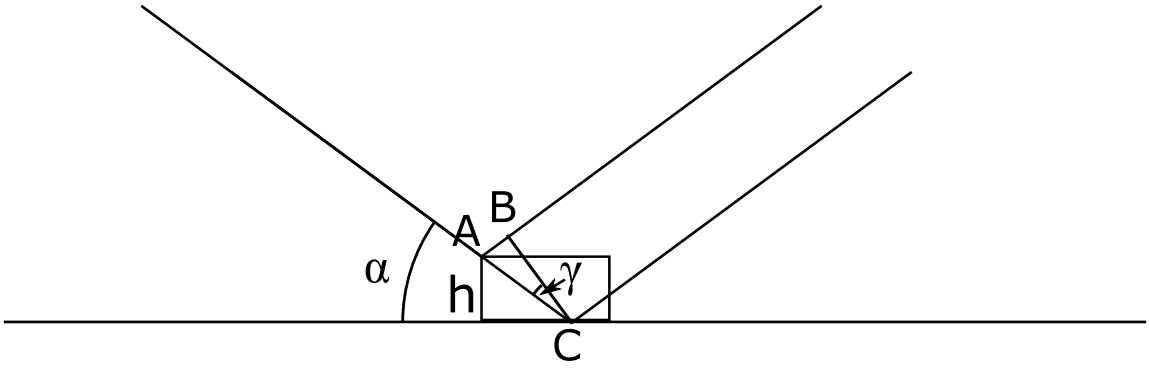

Figure 4: Effect of small height error on the reflected phase. The path length difference due to height error is $\overline{AB} - \overline{AC}$.

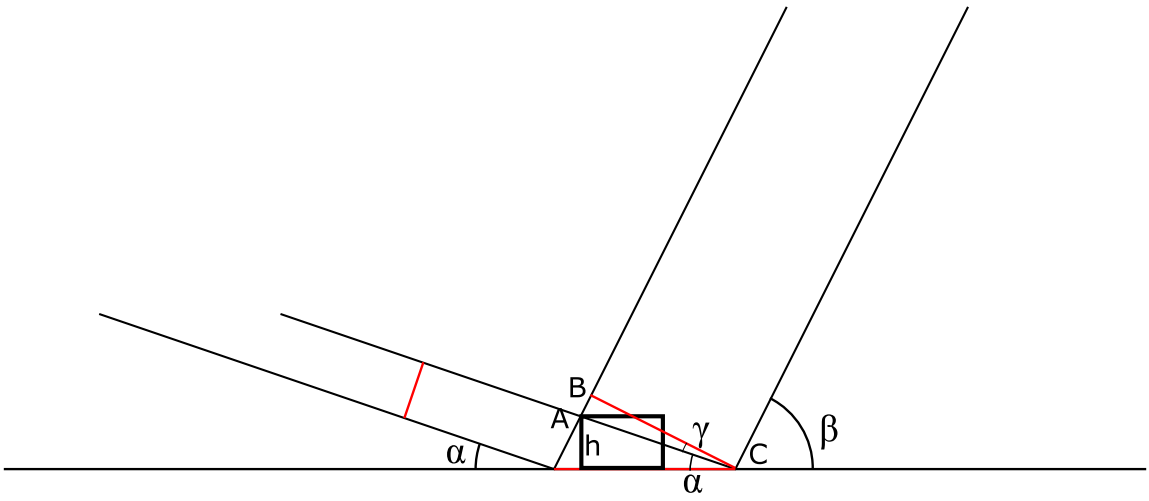

Figure 5: Effect of small height error on the reflected phase for a grating. The path length difference due to height error is $\overline{AB} - \overline{AC}$.