Lux Program

D. Sagan, K. Finkelstein

June 3, 2022

Contents

1 Introduction

Lux is a program for Monte Carlo simulations of X-rays using either coherent or incoherent ray tracing. Lux works by generating photons at a source element and tracking them through to a detector. Lux uses the Bmad subroutine library[1] for tracking and lattice bookkeeping.

Lux comes in two versions: A single threaded version whose executable is called `lux` and an parallel compute version using MPI whose executable is called `lux_mpi`. [Note to Distribution maintainers: The single threaded version will always be built when compiling the Distribution. The MPI version is built by building the Distribution with MPI enabled. See the documentation on the Bmad web site for more details.]

2 Running Lux

See the documentation for setting up the Bmad environmental variables at

https://wiki.classe.cornell.edu/ACC/ACL/RunningPrograms

Once the Bmad environmental variables have bee set, the syntax for invoking the single threaded version of Lux is

```
lux {-silent} {<master-input-file-name>}
```

The `<master-input-file-name>` optional argument is used to set the master input file name. The default is "`lux.init`".

The `-silent` optional argument suppresses the terminal output that Lux generates at the end of a run. This is useful to avoid long output when Lux is run repeatedly via a script.

Example input files are in the `$ACC_ROOT_DIR/lux/examples` subdirectory.

To run the parallel version of Lux, see the documentation on running programs on the Bmad web site for more details. If running on a single machine (as opposed to multiple machines in a cluster), typically `mpirun` or `mpiexec` is used. Example:

```
mpirun -np 8 lux_mpi
```

Note that two input files are always required to run Lux: The master input file (§4) and the Bmad lattice file (§5).

3 Fortran Namelist

Fortran namelist syntax is used for setting parameters in Lux's master input file (§4). The general form of a namelist is

```
&<namelist_name>
  <var1> = ...
```

```
  <var2> = ...
  ...
/
```

The tag "&<namelist_name>" starts the namelist where <namelist_name> is the name of the namelist. The namelist ends with the slash "/" tag. Anything outside of this is ignored. Within the namelist, an exclamation mark "!" and everything after it on a line is ignored. <var1>, <var2>, etc. are variable names. Example:

```
&section_def section =   0.0, "arc_std", "elliptical", 0.045, 0.025 /
```

here section_def is the namelist name and section is a variable name. Notice that here section is a "structure" which has five components (0.0, "arc_str", etc.).

4 Master Input File

4.1 Master Input File Example

The master input file consists of a single namelist called params. An example is:

```
&params
  lux_param%lattice_file = "test.bmad"         ! Defines experimental layout.
  lux_param%det_pix_out_file = "lux.det_pix#"
  lux_param%photon1_out_file = "lux.photon1"   ! Individual photon positions.
  lux_param%track_out_file = "track.dat"       ! Element-by-element track info.
  lux_param%bmad_parameters_out = ""           ! Parameters to include in
                                               !             det_pix_out_file.
  lux_param%photon_init_element = "p_init"     ! Name of element emitting photons
  lux_param%detector_element = "det"
  lux_param%photon1_element = ""

  lux_param%random_seed = 0                    ! 0 => Use system clock
  lux_param%random_engine = "pseudo"

  lux_param%scale_initial_field_to_1 = False
  lux_param%intensity_normalization_coef = 1e6 ! photon intensities norm.
  lux_param%normalization_includes_pixel_area = T

  lux_param%stop_total_intensity = 10          ! Cumulative intensity to stop at.
  lux_param%stop_num_photons = 1e8             ! Max number of photons to track.
  lux_param%mpi_run_size = 0.1                 ! Normalized num photons to track.

  lux_param%intensity_min_det_pixel_cutoff = 0.1 ! det_pix_out file Cutoff

  lux_param%intensity_min_photon1_cutoff = 1e-3  ! photon1_out file cutoff
  lux_param%n_photon1_out_max = 100              ! Max photons in photon1_out_file.
```

```
  lux_param%reject_dead_at_det_photon1 = False
  lux_param%timer_print_dtime = 300

  lux_param%histogram_variable = "x_angle"     ! Histogram independent var.
  lux_param%histogram_bin_width = 1e-4         ! Width of histogram bins.
/
```

4.2 Master Input File Parameters

The following is a complete list of the components of the params namelist:

`lux_param%bmad_parameters_out`

Semicolon delimited list of parameters from the Bmad lattice file to include in the `det_pix_out_file` header. This header records the values of various parameters and this can be used in post processing. For example, for the example Bmad lattice shown in Section §5.1, `lux_param%bmad_parameters_out` could be set to:

```
lux_param%bmad_parameters_out = "angle;cryst[graze_angle_in]"
```

In this case there are two parameters to be recorded in the `det_pix_out_file`: The parameter called `angle` and the parameter `cryst[graze_angle_in]`. This latter parameter is the `graze_angle_in` parameter of the `cryst` lattice element. In the `det_pix_out_file` header, Lux would put the values of these two parameters:

```
angle = 1.396263
cryst__graze_angle_in = 0.12634
```

Notice that the name used for the `cryst[graze_angle_in]` parameters has been mangled to make it readable by a python script. In particular, the "[" character is replaced by "__" and the "]" character is removed.

`lux_param%det_pix_out_file`

Name of the output data file recording the X-ray intensity on the detector. See section §12 for more details. The file name may be "numbered" using a "#" character (§9). A blank file name will prevent a file being generated.

`lux_param%detector_element`

Name of the lattice detector element where the photon are absorbed.

`lux_param%histogram_bin_width`

Bin width of histogram. See §12.3 for more details.

`lux_param%histogram_variable`

Variable used in constructing the histogram (§12.3). Possibilities are:

```
"init_x"        ! Initial photon x-position.
"init_y"        ! Initial photon y-position.
"init_x_angle"  ! Initial angle of photon velocity in the (z,x) plane.
"init_y_angle"  ! Initial angle of photon velocity in the (z,y) plane.
```

```
"x_angle"       ! Angle of photon velocity in then (z,x) plane at the detector.
"y_angle"       ! Angle of photon velocity in the (z,y) plane at the detector.
"energy"        ! photon energy relative to the reference (default).
```

If not given then no histogram file is produced.

`lux_param%intensity_min_det_pixel_cutoff`
: Sets the threshold absorbed photon intensity such that all pixels who's intensity is below this will not be included in the `det_pix_out` file (§12). This intensity is relative to the maximum pixel intensity.

`lux_param%intensity_min_photon1_cutoff`
: Cutoff intensity below which a photon will not be included in the photon1_out file (§11).

`lux_param%intensity_normalization_coef`
: Coefficient used to normalize the photon intensity with (§6).

`lux_param%lattice_file`
: Name of the input file that defines the X-ray optics setup from source to detector. The syntax of the file conforms to the Bmad lattice format. The Bmad manual has a detailed description of this format. Examples are presented in section §5.

`lux_param%mpi_run_size`
: Only used when running the parallel version of Lux. This parameter determines how many photons a slave process is asked to track before sending back data to the master process:

```
Number to track = lux_param%stop_num_photons *
          lux_param%mpi_run_size / (Number_processes - 1)
```

`%mpi_run_size` should be set to some number between 0 and 1. The value 0.1 is the default. Changing this number can affect running time but the running time should be fairly insensitive to the value of `%mpi_run_size`.

`lux_param%normalization_includes_pixel_area`
: Does the factor used for computing the normalized photon intensity (§6) include the pixel area of the detector?

`lux_param%photon_init_element`
: Name of the lattice element that is the source of the photons (§5).

`lux_param%n_photon1_out_max`
: Maximum number of photons to record in the `photon1_out_file`. Default is 100.

`lux_param%photon1_element`
: The photon position values in the photon1_out file will be the photon position at the lattice element named by `lux_param%photon1_element`. If `photon1_element` is blank (default), the detector element used will be used (§11).

`lux_param%photon1_out_file`
: Name of the output data file recording the starting and ending positions of individual photons. See section §11 for more details. The file name may be "numbered" using a "#" character (§9). A blank file name will prevent a file being generated.

`lux_param%random_engine`

Type of "random" number generator to use. Possibilities are:

```
quasi
pseudo      ! Default
```

The `quasi` number generator gives a more-or-less uniform distribution of numbers. That is, a quasi random generator is not actually random. The numbers generated with the quasi random generator are always the same from run to run.

`lux_param%random_seed`

If the `random_engine` is set to `pseudo`, `%random_seed` is the random number seed used by the random number generator. If set to 0, the system clock will be used so that the output results will show statistical variations from run to run. Not used if the `random_engine` is set to `quasi`. If the seed is nonzero, the results will be the same from run to run. However, the results when using the MPI version of Lux will not be exactly the same at the non-MPI Lux version.

`lux_param%reject_dead_at_det_photon1`

If `reject_dead_to_det_photon1` is set to True, do not print to the photon_out file any photons that do not reach the detector. This switch is only relevant when the lattice element at which the photon positions are being evaluated is not the detector (§11).

`lux_param%scale_initial_field_to_1`

Scale the field so that $E_x^2 + E_y^2 = 1$? Default is True. Only used if at least one field component is non-zero. See §6 for more details.

`lux_param%stop_num_photons`

Maximum number of photons to track (§7).

`lux_param%stop_total_intensity`

Cumulative unnormalized intensity at the detector to stop at (§7). Note: This parameter is ignored if running the parallel version of Lux and only `lux_param%stop_num_photons` is used.

`lux_param%timer_print_dtime`

The program will print a tracking status message every so often. The nominal time between status messages is set by `lux_param%timer_print_dtime` which is a number in seconds. The default is `300`.

`lux_param%track_out_file`

The `track_out_file` records how many photons are lost at each lattice element. This is useful for diagnosing cases when too many photons are being lost before reaching the detector.

5 Lattice Description File

The name of the lattice description file which defines the experimental setup is given by the `lattice_file` parameter in the master input file (§4). Bmad standard syntax is used[1].

The starting element for tracking photons, specified by the `lux_param%photon_init_element` parameter must be of type:

```
photon_init
```

This element may have an associated "physical source" element which should be a bend, wiggler, or undulator element.

The detector element, specified by the `lux_param%detector_element` parameter must be of type:

```
detector
```

The detector element must define a grid of pixels. Example:

```
det: detector, surface = grid = ix_bounds = (-100, 100),
                            iy_bounds = (-200, 200), dr = (0.001, 0.001)
```

See the Bmad manual for details of the detector syntax.

5.1 Example: Lattice Using an photon_init Source

An example lattice using an `photon_init` element without an associated physical source is:

```
1   beginning[e_tot] = 1e4
2   parameter[particle] = photon
3   parameter[no_end_marker] = T
4
5   c_rad = 75e-3  ! Crystal transverse radius
6   r0 = 0.2
7   angle  = 80 * pi / 180
8   qq = r0 * tan(pi/2-graze_angle)
9   dft_len = sqrt(qq^2 + r0^2)
10
11  src: photon_init, energy_distribution = gaussian
12  drift1: pipe, l = dft_len, x_limit = 0.01, y_limit = 0.01
13  cryst: crystal, crystal_type = 'Si(444)', b_param = -1, tilt = pi,
14          a2_trans_curve = 1 / (2 * c_rad), a4_trans_curve = 1 / (8 * c_rad^3)
15  drift2: drift, l = dft_len
16  det: marker, x_limit = 0.01, y_limit = 0.01
17
18  lux_line: line = (src, drift1, cryst, drift2, det)
19  use, lux_line
```

The list of lattice elements is given in line 19. This lattice constructs of five elements: The source, a crystal, and a detector with two drift spaces in between.

The definitions of the lattice elements is given in lines 11 through 17. The reference photon energy is specified in line 1 as 10 KeV and line 2 sets photons as the reference particle.

In this example the source element, called `src`, is specified on lines 11 and 12. The physical extent of the source is given by the parameters

```
l              ! Longitudinal length (2 σ)
x_half_length  ! Half length in x-direction (1 σ)
y_half_length  ! Half length in the y-direction (1 σ).
```

With Gaussian spatial distributions, `l` is the 2σ longitudinal length of the source and `x_half_length` and `y_half_length` are the 1σ transverse extents.

The source can be moved spatially by setting the parameters

```
x_offset, y_offset, z_offset
x_pitch, y_pitch, tilt
```

See the Bmad manual for more details.

5.2 Example: Lattice Using an Bend Source

When photons are generated via charged particles in an insertion device or a bend, the lattice must have (at least) two `branch` lines: One branch containing the insertion device or bend and the other branch is the X-ray branch containing the detector. Connecting the charged particle branch to the X-ray branch There must be a `photon_fork` element. Example Bmad lattice file:

```
1   beam_start[emittance_a] = 1.446E-7
2   beam_start[emittance_b] = beam_start[emittance_a] / 100
3
4   parameter[e_tot] = 5.289E+09
5   parameter[geometry] = open
6
7   beginning[beta_a]  =  2.2611
8   beginning[alpha_a] = -1.35379
9   beginning[eta_x]   =  0.549325
10  beginning[etap_x]  = -0.064612
11
12  beginning[beta_b]  = 9.3144
13  beginning[alpha_b] = 1.12414
14  beginning[eta_y]   = 0
15  beginning[etap_y]  = 0
16
17  b05w: sbend, l = 3.237903, angle = 0.102289270 ! rho =  31.65434
18
19  bend_line: line = (b05w)
20  c_fork: photon_fork, to_line = c_line, superimpose,
21          ref = b05w, ref_origin = beginning, offset =  23.6980 - 23.3814
22
23  !----------------------------
```

```
24
25  drift1: drift, l = 10.25
26  drift2: drift, l = 10.75 - drift1[l]
27  drift3: drift
28  drift4: drift
29
30  white_beam_slit: rcollimator, x_limit = 0.005/2, y_limit = 0.001/2
31  mono_slit: rcollimator
32
33  cryst1: crystal, crystal_type = Si(531), ref_tilt = -pi/2, b_param = -1
34  cryst2: crystal, crystal_type = Si(531), ref_tilt = pi/2, b_param = -1
35
36  det: detector, x_limit = 1, y_limit = 1,
37       surface = {grid = {ix_bounds = (-1, 1), iy_bounds = (-1, 1), dr = (1, 1)}}
38
39  c_line: line = (drift1, white_beam_slit, drift2, cryst1, drift3,
40                  cryst2, drift4, mono_slit, det)
41  c_line[particle] = photon
42  c_line[e_tot] = 8.979e3
43
44  use, bend_line
45
46  expand_lattice  ! So that bragg angles are computed
47
48  ! drift3 length is set so that crystal2 is positioned 75 mm
49  ! vertically from the beam plane
50
51  drift3[l] = 0.075 / sin(cryst1[bragg_angle_in] + cryst1[bragg_angle_out])
52  drift4[l] = 14.5 - (drift1[l] + drift2[l] + drift3[l])
```

Lines 1 through 21 define the charged particle branch which is simply a bend called `b05w` with a `photon_fork` element, called `c_fork`, superimposed on top of it.

The `c_fork` element branches to a branch called `c_line`. The elements of `c_line` are given in lines 39 and 40: Two slits, two crystals, and a detector with drifts in between.

6 Photon Description

Photons are described by:

```
(x, vx/c, y, vy/c, z, vz/c)   ! six dimensional phase space vector
E                             ! Photon energy (eV)
e_field_x, e_field_y          ! Electric field magnitude vector
phase_x, phase_y              ! Electric field phase
```

(x, y, z) is the spatial position of the photon with z being the longitudinal coordinate and $z = 0$ corresponds to the beginning of the element the photon is passing though (Thus z is generally not interesting). $(v_x/c, v_y/c, v_z/c)$ is the photon velocity normalized to 1. Generally photons that make it to the detector will have v_z/c close to 1 and v_x/c and v_y/c small.

The photon phases `phase_x` and `phase_y` are only relevant with coherent tracking set in the lattice by the `parameter[photon_type]`. See the Bmad manual for more details.

The unnormalized intensity of a photon is:

```
I(unnormalized) = e_field_x^2 + e_field_y^2
```

Initially, the intensity of a photon is set to 1 if `lux_param%scale_initial_field_to_1` is set to True (the default). Photons can loose intensity, for example when diffracting off of a crystal.

To properly normalize the light intensity of the detector pixels in the output files, a normalization factor f_n is applied.

$$I(\text{normalized}) = f_n I(\text{unnormalized}) \tag{1}$$

If `lux_param%normalization_includes_pixel_area` is set to True (the default), f_n is constructed so that the normalized intensity will be independent of the number of simulation photons generated and independent of the detector pixel size. Explicitly:

$$f_n = \frac{C_f}{A_{pix}\, N_p} \tag{2}$$

where A_{pix} is the area of a pixel, N_p is the number of photons tracked, and C_f is a normalization coefficient set by the `intensity_normalization_coef` parameter in the master input file, C_f can be used, for example, to scale the output numbers to correspond to actual measured values. If `lux_param%normalization_includes_pixel_area` is set to False, f_n in Eq. (2) is constructed to only normalize the number of photons generated:

$$f_n = \frac{C_f}{N_p} \tag{3}$$

6.1 Photon Polarization Initialization

When using a `photon_init` element without an associated "physical element", the polarization of the photons is set by the `photon_init` parameters `e_field_x` and `e_field_y`.

When using a `photon_init` element with an associated physical element (which is generally a bend, wiggler, and undulator element), the polarization the photons is determined by the properties of the physical element.

If the parameter `lux_param%scale_initial_field_to_1` is set to True, Lux will scale the field so that the `unnormalized` intensity (§6) of each photon is 1 at the start of tracking. If both field components are set to zero, the polarization will be random. As photons travel, they can loose intensity via, for example, diffraction from a crystal.

7 Photon Detection

Photons are tracked until they are lost (hit an aperture) or until they get to the detector which is the last element in the lattice.

Two parameters in the master input file determine when the simulation is stopped:

```
lux_param%stop_total_intensity
lux_param%stop_num_photons
```

If `lux_param%stop_total_intensity` is positive, the simulation ends when the total accumulated unnormalized intensity at the detector passes this threshold. Intensity is the square of the photon electric field and the unnormalized photon intensity will vary from zero to one for each photon.

If `lux_param%stop_num_photons` is positive, the simulation is stopped when the number of photons generated passes this threshold. If both stop parameters are positive, the simulation is stopped when either the intensity or the number of photons passes the set threshold.

8 Output Data Files

The names of the output data files are specified by the following parameters set in the master input file (§4):

```
photon1_out_file    ! §11
det_pix_out_file    ! §12
track_out_file      ! §10
```

Note that the `det_pix_out_file` name is used as a prefix to generate four different data files.

9 Numbering the Output Data Files

If an output data file name (see above) contains a pound character "#", then a number will substituted for the pound character and the number substituted will be increased by one each time Lux is run. For example, if `det_pix_out_file` is defined by

```
lux_param%det_pix_out_file = 'lux.det_pix#'
```

Then the first time Lux is run, the pixel data file will be named "lux.det_pix1". The next time Lux is run, the data file will be named "lux.det_pix2", etc.

Using a numbering system prevents data files from being overwritten. If more than one output file name has a pound character, all such files will receive the same number on a given run. To set the number, one can edit the file:

```
lux_out_file.number
```

This file is created by Lux the first time Lux is run in a directory and then, every time there after Lux is run, Lux will increment the number contained in this file.

When using the parallel version of Lux, if the `photon1_out_file` contains an "" character, then, for each slave process doing tracking, that slave process will generate a file with the rank id number of slave process substituted for the character. If the single tread version of Lux is being used then a zero character will be substituted.

10 Track Data File

The `track_out_file` records how many photons are lost at each lattice element. This is useful for diagnosing cases when too many photons are being lost before reaching the detector.

11 Photon1 Data File

The `photon1_out_file` parameter (set in the master input file) is the name of an output data file containing beginning photon coordinates and the photon coordinates at the lattice element specified by `lux_param%photon1_element`. If `lux_param%photon1_element` is blank, the end coordinates are evaluated at the detector element.

If `reject_dead_at_det_photon1` is set to True, photons that do not make it to the detector will not be included in the photon1_out file. In any case, a photon that does not reach the `photon1_element` element will not be included. Thus, if the lattice element where the photon position is being evaluated at the detector, it will always be the case that photons that do not make it to the detector will not be included in the photon1_out file irregardless of the setting of `reject_dead_at_det_photon1`.

If the file name is blank then no file will be generated. The file name may be "numbered" using a "#" character (§9).

If a photon at the detector has an intensity of less than `lux_param%intensity_min_photon1_cutoff` then the photon is not counted.

Each row in the file is a single photon. The columns in the file are:

```
n_live           ! Index of this photon.
beginning_orbit  ! Five columns: (x, vx, y, vy, z) at the source
orbit_at_ix_ele  ! Five columns: (x, vx, y, vy, z) at the ''end''
energy           ! Photon energy (eV).
intensity_x      ! x-axis unnormalized photon intensity
intensity_y      ! y-axis unnormalized photon intensity
```

12 Detector Pixel Data Files

The `det_pix_out_file` parameter (set in the master input file) is a prefix used to form the names for four output data files. If the file name is blank then no files will be generated. The file name may be "numbered" using a "#" character (§9). The names of the four files are:

```
<det_pix_out_file>            ! Pixel-by-pixel file (§12.1).
<det_pix_out_file>.x          ! X-projection file (§12.2).
<det_pix_out_file>.y          ! Y-projection file (§12.2).
<det_pix_out_file>.histogram  ! Histogram file (§12.3).
```

Each file contains a table of data. All of the files have the following columns (called the "common" columns):

```
intensity_x       ! Normalized pixel intensity of x-polarized light.
intensity_y       ! Normalized pixel intensity of y_polarized light
intensity         ! Normalized pixel intensity = intensity_x + intensity_y
n_photon          ! Number of photons hitting pixel (or bin with histogram data).
E_ave             ! Average energy deviation from reference (eV)
E_rms             ! RMS about the average energy (eV)
x_ang_ave         ! Average angle in the x-z plane of the striking photons.
x_ang_rms         ! RMS of angle of the striking photons in the x-z plane.
y_ang_ave         ! Average angle in the y-z plane of the striking photons.
y_ang_rms         ! RMS of angle of the striking photons in the x-z plane.
init_x_ang_ave    ! Initial photon average angle in the x-z plane.
init_x_ang_rms    ! Initial photon RMS of angle in the x-z plane.
init_y_ang_ave    ! Initial photon average angle in the y-z plane.
init_y_ang_rms    ! Initial photon RMS of angle in the x-z plane.
```

All averages (`E_ave`, `x_ang_ave`, etc) are intensity weighted averages. `E_ave` is the photon energy deviation from the reference energy. `E_rms` is the RMS variation of the photon energy with respect to `E_ave`.

12.1 Pixel-By-Pixel File

The first data file has the `det_pix_out_file` parameter as the file name. This file contains pixel-by-pixel information. The intensity of a pixel is the accumulated intensity of the photons hitting the pixel. Only pixels whose intensity is non-zero and whose intensity is greater than `lux_param%intensity_min_det_pixel_cutoff` are recorded in the output file. This cutoff is relative to the intensity of the pixel with the highest intensity. `lux_param%intensity_min_det_pixel_cutoff` is useful for keeping the data file sizes small. Each row in this file represents a single pixel. The columns in this file are:

```
ix_pix            ! $x$-axis index
iy_pix            ! $y$-axis index
x_pix             ! $x$ coordinate of pixel center
y_pix             ! $y$ coordinate of pixel center
... and 14 common columns ..
```

12.2 X and Y Projection Files

The second and third data files contain data projected on the x and y detector axes. The name of these two files will be the same as `det_pix_out_file` with ".x" and ".y" suffixes indicating projected data on the x and y axes respectively. The columns for the x-axis projected data is:

```
ix_pix            ! x-axis index
x_pix             ! x coordinate of pixel center
... and 14 common columns ..
```

With similar columns for the y-axis file. The columns for the x-axis projected file records the sum of the intensities for all pixels with a common `y_pix`. Since this can include pixels not recorded in the pixel-by-pixel file (due to a non-zero `lux_param%intensity_min_det_pixel_cutoff` setting), quantities like the total number of photon need not match between the pixel-by-pixel file and the projected files.

12.3 Histogram File

The fourth data file will have the same name as `det_pix_out_file` with a ".histogram" suffix. This file contains This file will contain a table with columns:

```
histogram_variable
... and 14 common columns ..
```

Each row in the file represents one bin.

The photon property to be histogrammed is given by `lux_param%histogram_variable` which may be one of:

```
"init_x"        ! Initial photon x-position.
"init_y"        ! Initial photon y-position.
"init_x_angle"  ! Initial angle of photon velocity in the (z,x) plane.
"init_y_angle"  ! Initial angle of photon velocity in the (z,y) plane.
"x_angle"       ! Angle of photon velocity in then (z,x) plane at the detector.
"y_angle"       ! Angle of photon velocity in the (z,y) plane at the detector.
"energy"        ! photon energy relative to the reference (default).
```

`"energy"` is the default.

The width of the histogram bins is set by `lux_param%histogram_bin_width`. The value given for a histogram bin is computed using the formula:

```
intensity_of_photons_in_bin / bin_width
```

All photons hitting any detector pixel are used for the histogram.

13 Plotting

13.1 Data Visualization of the det_pix_out_file

There is a python script for making an intensity plot of the `det_pix_out_file` in:

```
$ACC_ROOT_DIR/lux/scripts/plot_det_pix.py
```

The script needs python3 to run. You can invoke the script using the command:

```
<path-to-lux-root-dir>/scripts/plot_det_pix.py <det_pix_out_file>
```

where `<det_pix_out_file>` is the actual name of the `det_pix_out_file`. If the default version of python is not python3 then the following may work

```
python3 <path-to-lux-root-dir>/scripts/plot_det_pix.py <det_pix_out_file>
```

The script has a number of optional arguments. The syntax for running the script is:

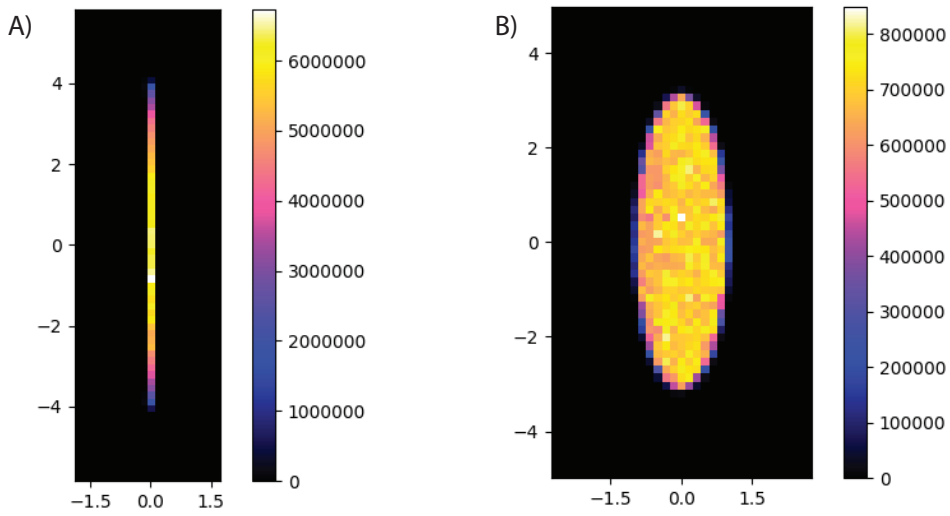

Figure 1: Example intensity plots. A) Results from a perfectly configured Rowland circle spectrometer. B) Same as (A) except that the curvature of the crystal is off by 1%. The lattice used to generate these plots is illustrated in the Bmad manual in the `Lattice Examples` chapter.

```
plot_det_pix.py {-aspect <aspect_ratio>} {-scale <scale>}
                                          {-plot <who_to_plot>} {<data_file_name>}
<who_to_plot> = x         # Intensity of x-polarized photons
              = y         # Intensity of y-polarized photons
              = i         # Total intensity (sum of x & y polarizations)
              = e         # Energy
Defaults:
  <aspect_ratio>   = 1
  <scale>          = 1e3
  <data_file_name> = det_pix
  <who_to_plot>    = i
```

Figure 1 shows two example intensity plots.

The `<scale>` parameter sets the scale of the axis numbers. A scale of `1e3` results in the axis numbers being in millimeters.

The `<aspect_ratio>` parameter sets the aspect ratio of the plot. An aspect ratio of one (the default) will mean that the vertical and horizontal scales (detector position per plotted unit length) begin the same. An aspect ratio greater than one will enlarge the vertical scale and less than one will enlarge the horizontal scale.

13.2 Other plotting

For lattice elements like crystals or mirrors that can have a curved surface, there is a script for drawing the surface in the directory:

`util_programs/photonic_surface_plot`

In this directory there is a `README` file with documentation.

14 Python Scripting

In some cases it is desired to study some output parameter while varying some input parameter. For example, it may be desired to look at a "rocking curve" where the initial angle of the photons are varied and the output intensity is monitored. Such studies can be easily achieved using a scripting language like `python`.

Example:

```
#!/usr/bin/env python

import subprocess
import glob

# open output file

out_file = open('output.dat', 'w')
out_file.write('#       Var         I_x/I         I_y/I   (I_x-I_y)/I           I_x

# Loop over all runs

n_max = 10
dvar = 0.00001
for ix in range(-n_max, n_max+1):

  # Create file with var_to_change set to correct value

  var = ix * dvar
  print ('Run with my_var set to: ' + str(var))

  b_file = open ('var.bmad', 'w')
  b_file.write('my_var = ' + str(var))
  b_file.close()

  # Run lux
  # Remove any Bmad "digested" files to make sure Bmad rereads the lattice file.

  if len(glob.glob('*digested*')) > 0: subprocess.call('rm *digested*', shell=True)
  subprocess.call('/home/dcs16/linux_lib/production/bin/lux -silent', shell=True)

  # Get output
```

```
33
34      with open('det.pix', 'r') as d_file:
35        for line in d_file:
36          if 'intensity_x_norm' in line: exec(line)
37          if 'intensity_y_norm' in line: exec(line)
38          if 'intensity_norm' in line: exec(line)
39
40      # write results
41
42      ii = intensity_norm
43      ix = intensity_x_norm
44      iy = intensity_y_norm
45
46      if ii == 0:
47        out_file.write('{:12.2e}'.format(var) + '{:14.4e}'.format(0) +
48                       '{:14.4e}'.format(0) + '{:14.4e}'.format(0))
49      else:
50        out_file.write('{:12.2e}'.format(var) + '{:14.4e}'.format(ix/ii) +
51                       '{:14.4e}'.format(iy/ii) + '{:14.4e}'.format((ix-iy)/ii))
52
53      out_file.write('{:14.4e}'.format(ix) + '{:14.4e}'.format(iy) +
54                     '{:14.4e}'.format(ii) + '\n')
55
56    out_file.close()
```

The heart of the script is the loop that begins on line 15. Each iteration of this loop creates a file called `var.bmad` (lines 22 through 24) with one line setting the variable `my_var` to a value from `-n_max * dvar` to `n_max * dvar` which in this case is from -10e-5 to 10e-5. The script then runs Lux (line 30), and then extracts the pertinent data (lines 34 to 38). Finally at the end of the loop the data is appended to the output file in lines 42 through 54.

The lattice file that Lux uses is called `lat.bmad` (this is set by the `lattice_file` parameter in the `lux.init` file which is not shown). This file contains the following:

```
call, file = var.bmad   ! Read in my_var variable
cryst1: crystal, ..., x_pitch = my_var
```

The `my_var` variable sets the `x_pitch` attribute of the crystal named `cryst1`.

To get the data after Lux is run, the script opens up a file `det.pix`. This name has been set in the `lux.init` file as the name of the `det_pix_out_file` parameter:

```
det_pix_out_file = 'det.pix'
```

The `det_pix_out_file` has a number of initial rows that give overall parameters. Example:

```
master_input_file   = "lux.init"
lattice_file        = "lat.bmad"
```

```
normalization       =  1.000000E+02
intensity_x_unnorm  =     9.41577E+01
intensity_x_norm    =     9.41577E+03
intensity_y_unnorm  =     4.64621E-02
intensity_y_norm    =     4.64621E+00
intensity_unnorm    =     9.42042E+01
intensity_norm      =     9.42042E+03
```

When the script (see line 46) finds a line in `det.pix` that has the string `intensity_x_norm`, the script "executes" this line. The result is that the script set a variable named `intensity_x_norm` to the value given in the `det.pix` file. Similarly, the script extracts the values of `intensity_y_norm` and `intensity_norm` in lines 47 and 48.

References

[1] D. Sagan, "Bmad: A Relativistic Charged Particle Simulation Library" Nuc. Instrum. & Methods Phys. Res. A, **558**, pp 356-59 (2006).