wiscobolt (implementation)

Note: This document is a work-in-progress

Muhsin H. Younis
Department of Medical Physics
University of Wisconsin − Madison
Madison, Wisconsin

First published 10-21-2023
Latest version: 0.1, published 10-21-2023

Table of Contents

Introduction

In this document, we hope to describe:

- **Section 1**: How to use wiscobolt, i.e., how to create your own input file Additionally, the program outputs, i.e., how to interpret the values produced by wiscobolt as well as how raw data files are formatted for your own use outside of wiscobolt.
- (WIP): Implementation of the various algorithms and numerical methods described in the main **wiscobolt** document, as well as those which are encountered throughout the course of creating a Boltzmann transport solver that were not mentioned in the main document. These sections go module-to-module, describing the purpose and implementation of particular subroutines or sets of subroutines.

1 User Input

The user's goal is to specify the geometry, beam, and energy regime of their problem, in addition to the discretization methods, discretization parameters, and solution methods they wish to use, and finally make certain choices as to the interaction physics and other options available.

The input file is formatted as follows: Headers are denoted with three hashes on either side of the header name, as an example, `### Solution parameters ###`. Subheaders are denoted with one hash before the subheader name, such as `# Problem`. Subheaders with an asterisk, such as `#* Photon solution method`, are non-mandatory inputs, and may be specified but will only be required to be specified by the user under certain conditions, as we will thoroughly describe soon. But as an example for the aforementioned subheader, if the user defines their problem as, say, `external electron beam`, then it will not matter what their input is for `#* Photon solution method` (but the input must still be among the valid options, such as `SI` or `GMRESm`). Finally, additional inputs whose structure will depend on the choice for the previous input are denoted with `#-->`. For example, `#--> Max number of iterates`, which is actually required if either `SI` or `GMRESm` is used, but requires one input for the former and two for the latter.

Now, we will visit each header, and subheader, and describe the meaning of the inputs. It is recommended that the user follows along with the `example_input.txt` that is included in the `Input files` folder in the main directory. This example input describes an external photon beam problem in a water tank mesh that was generated by the author using the program Gmsh.

### Solution parameters

# Problem

Type: character
Options: 3

→ `external photon beam`: Transport of photons only, which are sourced by an external beam. Note that, while this problem involves only photon transport, scattering mechanisms that produce electrons, such as the photoelectric effect, are still involved in attenuation of photons. However, these created electrons will not subsequently produce photons via electron scattering mechanisms which produce photons, such as Bremsstrahlung.

→ `external photon beam coupled`: An external photon beam, but both photon and electron transport are performed, the two being coupled via photons creating electrons, *but not* electrons creating photons. This means neglect of Bremsstrahlung photons, or if the user is interested in energy deposition (as will be described later on), deposition of Bremsstrahlung energy locally.

→ `external electron beam`: Transport of electrons only. Similar to `external photon beam`, photons produced by electrons during transport do not produce more electrons that are considered during computation.

# Use FCS

Type: logical

The user can specify whether or not to use the first collision scattering source method [1]. If using `FEXS` energy discretization (described later), FCS will automatically be turned off regardless of user input.

#* [Photon/Electron] solution method

Type: character
Options: 2

→ `SI`: Source iteration. Strongly recommended for photons, strongly not recommended for electrons. Note that this option *can not* be used with the `PN` option which we will later describe in the context of angular discretization.

`#--> Max number of iterates`: **Type: integer**. The total number of iterates allowed for source iteration. If this number is exceeded during iteration without convergence being reached, the solver will take the most recent iterate as the solution.

→ `GMRESm`: Generalized minimal residual method with restart. Strongly recommended for electrons, acceptable for photons but not necessarily recommended.

`#--> Max number of iterates`: **Type: integer [2]**. The first entry is the number of iterates inbetween ‘restarts.’ Note that a higher number *will* lead to larger storage requirement, however, its affect on speed is not as simple. The second entry is the total number of iterates after which the solver will proceed to the next energy group using the most recent iterate as the solution.

# Convergence criterion

Type: real

The convergence criterion is ϵ. For `SI`, we iterate until such an iterate p that:

$$\epsilon > \frac{|\varphi^p - \varphi^{p-1}|_2}{|\varphi^{p-1}|_2} \tag{1.0.1}$$

Where φ^p is the fluence of particles that have scattered up to p times, $|x|_2$ is the L2-norm of the vector x. Thus, if you take, say, $\epsilon = 10^{-8}$, you are seeking the iterate such that the fluence of particles which have scattered exactly p times is on the order of 10^{-8} times as much as the total fluence up to $p-1$ times, indicating that your iterates are only incrementing the total solution φ^p by negligibly little. For `GMRESm`, the convergence criterion is:

$$\epsilon > |s - \hat{L}\psi^p|_2 \text{ or } \epsilon > |\hat{T}^{-1}s - \hat{T}^{-1}\hat{L}\psi^p|_2 \tag{1.0.2}$$

Where s is the source, $\hat{L}$ is the BTE operator, and ψ^p is now defined as the angular fluence solution following iterate p (though iterates like ψ^p in GMRESm don't have the same physical meaning as those in SI). The quantity $|s - \hat{L}\psi|_2$ is simply the residual, so you are asking for the residual to be lower than ϵ. The 'or' in the equation above is more or less of a technicality as far as the user is concerned, as in P_N, we take the problem to be $\hat{L}\psi = s$, but in S_N, we actually take the problem to be $\hat{T}^{-1}\hat{L}\psi = \hat{T}^{-1}s$.

### Mesh

This section requires a discussion first. Currently, wiscobolt imposes the following restrictions on the mesh created and provided by the user:

- The scale is in centimeters.
- The mesh must be created by Gmsh (`https://gmsh.info/`), formatted as per Version 4.1. Note the distinction between 'geometry,' and a 'mesh,' with which I will be very particular throughout this section. Gmsh asks the user to create a geometry before creating a mesh, and all of the information that Gmsh outputs is important in wiscobolt; both the geometry and the mesh.
- The mesh must have only tetrahedral elements.
- The geometry must specify at least one 'Physical Volume.' These physical volumes are used by wiscobolt to specify materials as well as the beam (though a beam isn't necessary for the user to specify). It is absolutely necessary to have at least one material, which is designated by the 'Physical Tag' being 1, but the name doesn't matter. Subsequent materials are specified as 2, 3, etc., and then the user must use the input file to tell wiscobolt *what* materials correspond to these numbers (we'll describe how later on). If a user chooses to specify their beam, then they must use the physical tag 2001 for it.

- The whole volume may not be concave, and may not have cavities. If the user really needs to solve problems in such a context, there is a rather slow and wasteful workaround that is technically implemented, which involves specifying *vacuum* in their materials input, which is assigned to parts of the volume that exist to make it technically convex, and cavities, so that the entire geometry is in fact convex but particle transport through vacuum elements will be treated appropriately. If this is how you want to do it, my advice is to very sparsely mesh the vacuum geometry pieces.
- For convenience, the mesh may be translated and scaled within wiscobolt, this will be described later on.

# Selected mesh

Type: character

The user must specify the folder name of the mesh they wish to use. The folder must be placed within the `Mesh files` folder in the main directory. The mesh file within the folder must be named `mesh.msh`. See the beginning of this section for the restrictions that the mesh must satisfy to be used in wiscobolt. It is recommended, but not required, that the user also keep their Gmsh geometry file as well as a text file describing the geometry and mesh, such as the size of the geometry, whether or not a beam is specified, what the beam cutout may or may not be, number of materials, number of elements, etc. An example mesh is included with wiscobolt, `GMSH_cube_pyramid-beam_9248_46821`. The user can look at the mesh file with any text editor.

# Beam in mesh

Type: logical

The user must specify `true` if their mesh contains a physical volume (tagged 2001) which describes the beam in the mesh. The user must specify `false` otherwise. Note that if the user intends for their whole geometry to be engulfed in the beam (relevant input to be described later), they could or could not specify the beam physical volume in Gmsh, it actually makes no difference in this special case.

It is recommended in general for the user to specify their beam in the mesh, not for wiscobolt's convenience (as the beam is still specified by user input), but rather because the presence of a particularly shaped beam will lead to steep spatial falloff of the uncollided fluence at the elements on the edge of the beam. Without providing a specific shape for the beam within the geometry and thus the mesh, their beam may have a 'jagged' shape and consequently actually describe more source particles than actually exist in the problem.

### Angular discretization

#* [Photon/Electron] solution method

Type: character

Options: 2

→ `SN`: Discrete ordinates. That is, the problem references the angular fluence at precise, discrete 'ordinates' as the quantities of interest. The ordinates are chosen separately along the polar angle cosine μ and azimuthal angle ϕ. These ordinates are particularly chosen in accordance with a Gauss-Legendre and Gauss-Chebyshev quadrature set, respectively.

→ `PN`: Real spherical harmonics expansions. That is, the problem references real spherical harmonic moments of the angular fluence as the quantities of interest. The conventions for the real spherical harmonics are given as follows. First, we use spherical harmonics:

$$Y_\ell^m(\hat{\mathbf{k}}) = \sqrt{\frac{(2\ell+1)}{4\pi}\frac{(\ell-m)!}{(\ell+m)!}} P_\ell^m(\mu) e^{im\phi} \tag{1.0.3}$$

with the Condon-Shortley phase present in the associated Legendre polynomials $P_\ell^m(\mu)$, which are given by:

$$P_\ell^m(\mu) = (-1)^m (1-\mu^2)^{m/2} \frac{d^m}{d\mu^m}\big[P_\ell(\mu)\big] \tag{1.0.4}$$

and then $P_\ell(\mu)$ is a Legendre polynomial, given by:

$$P_\ell(\mu) = \frac{1}{2^\ell \ell!}\frac{d^\ell}{d\mu^\ell}\big[(\mu^2-1)^\ell\big] = P_\ell^0(\mu) \tag{1.0.5}$$

Finally the real spherical harmonics are given by:

$$y_\ell^m(\hat{\mathbf{k}}) \equiv \begin{cases} \frac{1}{\sqrt{2}}\big[Y_\ell^{-m}(\hat{\mathbf{k}}) + (-1)^m Y_\ell^m\big], & m > 0 \\ Y_\ell^0(\hat{\mathbf{k}}), & m = 0 \\ \frac{i}{\sqrt{2}}\big[Y_\ell^m(\hat{\mathbf{k}}) - (-1)^m Y_\ell^{-m}(\hat{\mathbf{k}})\big], & m < 0 \end{cases} \tag{1.0.6}$$

all with $-\ell \leq m \leq \ell$ and $\ell \geq 0$. The spherical harmonic moments are indexed in wiscobolt and in output files by $q = \ell(\ell+1) + m + 1$, which maps uniquely to (ℓ, m) given the aforementioned restraints on (ℓ, m). Specifically, $\ell(q) = \lceil\sqrt{q}\rceil - 1$ and $m(q)$ follows from rearranging $q(\ell, m)$ with this $\ell(q)$. The index q essentially counts spherical harmonics from $(0,0)$, to $(1,-1)$, $(1,0)$, $(1,1)$, $(2,-2)$, and so on.

Note finally that `PN` *can not* be used with the `SI` solution method. This is a permanent feature, as the use of `SI` with `PN` is needlessly counter-productive, given the specific advantages of `SI`, specifically the so-called 'sweep' (see the main **wiscobolt** document).

As of the 0.1 version of wiscobolt, S_N is the only available method (this is because the P_N method is appearing to require excessively large amounts of storage, and so it remains a work-in-progress developer option only).

# Angular discretization parameter

Type: integer

This parameter will describe the refinement of angular discretization.

The S_N method is in principle described by: N_μ (the number of polar discrete ordinates) and N_ϕ (the number of azimuthal discrete ordinates). Additionally, however, the S_N method always uses what is referred to as 'P_N scattering,' for which it is necessary to also supply L (the max Legendre moment used in the scattering treatment). Now, for reasons related to quadrature, it may be recommended if not required that these parameters are related to one another as:

$$N_\mu = \frac{1}{2}N_\phi = L + 1 \tag{1.0.7}$$

So, the user-defined 'angular discretization parameter,' say A, is actually N_μ, which will then be related to N_ϕ and L as above.

For the P_N method, L is the only required parameter. However, if the user wishes to convert from P_N to S_N, they *must* use angular quadrature with N_μ and N_ϕ described by the constraint above, for the given L. The reason for this is well described in the main **wiscobolt** document. Thus, for good measure, we still take L from the angular discretization parameter with $L = A - 1$. It is worth noting that since $\ell = 0, ..., L$, $A = L + 1$ does describe the *total* number of Legendre moments (but not the *max* number).

### Energy discretization

# Energy discretization method

Type: character

Options: 2

→ `MGXS`: Uses multigroup Legendre formalism. That is, the method assumes that, over the gth energy 'group' such as $E_{g+1} \leq E \leq E_g$, the angular fluence and source are constant. The quantities of interest become, for instance:

$$\psi_g \equiv \int_{E_{g+1}}^{E_g} dE \ \psi(E) \tag{1.0.8}$$

While a constant $\psi(E)$ leads to $\psi\Delta E_g$, it is essentially an acknowledgement of the approximate nature of this approximation to use the above more generally. Taken over $\Delta E_g = E_g - E_{g+1}$, you obtain the average angular fluence in the group g, which is essentially what we are assuming to be the constant group fluence. Note that a linear variation between average fluences defined at group midpoints could possibly be taken after calculation, but is not actually faithful to the approximation, so in wiscobolt any calculations following solution are actually worked through with fluences that are truly constant within groups.

→ `FEXS`: (first order only) Uses finite element Legendre formalism. Application of the 1D finite element method to energy discretization has been proposed, as far as the author is aware, by the creators of SCEPTRE [2]. The method assumes that, over the gth energy group, the angular fluence varies linearly with energy:

$$\psi(E) = \frac{E - E_{g+1}}{\Delta E_g}\psi(E_g) + \left(1 - \frac{E - E_{g+1}}{\Delta E_g}\right)\psi(E_{g+1}),\ E_{g+1} \leq E \leq E_g \tag{1.0.9}$$

As it is implemented in wiscobolt, it also assumes that the source spectrum varies linearly with energy (but, this is not a fixture of the finite element method in this context, it is only a simplification). Notably, FEXS can not be used with FCS, so if the user turns on FCS with FEXS, wiscobolt will turn it off.

Note that FEXS currently incurs between two to four times as much computation times and storage requirements as MGXS.

Discretization of cross sections is described in **wiscobolt physics**. Discretization of sources deserves a discussion at the end of this section, as decisions for the following discretization parameters must be made with respect to the beam's energy spectrum.

#* [Photon/Electron] energy min and max

Type: real [2]

The first entry is the minimum energy in MeV, the second entry is the maximum energy in MeV. Currently, wiscobolt does not go below 10 eV, nor above 100 MeV.

The phrase 'minimum energy' means that, below this energy, we assume that any particles created are not necessary to be transported. 'Maximum energy' means that this is the energy of the most energetic particles in the source spectrum. This is because particles will never gain energy via scattering, so we won't need to describe higher energies than this.

#* Number of [photon/electron] energy groups

Type: integer

The number of energy groups to use for the given particle. Note that, for G groups, there are $G + 1$ total energy 'nodes.'

#* [Photon/Electron] energy structure

Type: character
Options: 3

→ `linear`: ΔE_g is kept roughly constant. For G groups, with min and max energies $E_{\min}$ and $E_{\max}$, the gth energy node is given by:

$$E_g = \frac{1}{G}(E_{\min} - E_{\max})(g - 1) + E_{\max} \tag{1.0.10}$$

→ `logarithmic`: ΔE_g is larger for higher energies and smaller for lower energies. The quantity $\ln(E_g/E_{g+1})$ is constant. The gth node is:

$$E_g/\text{MeV} = \exp\left[\frac{1}{G}\ln\left(\frac{E_{\min}}{E_{\max}}\right)(g-1) + \ln(E_{\max}/\text{MeV})\right] \tag{1.0.11}$$

→ `exponential`: ΔE_g is smaller for higher energies and larger for lower energies. The quantity $e^{E_g/\text{MeV}} - e^{E_{g+1}/\text{MeV}}$ is constant. The gth node is:

$$E_g/\text{MeV} = \ln\left[\frac{1}{G}(e^{E_{\min}/\text{MeV}} - e^{E_{\max}/\text{MeV}})(g-1) + e^{E_{\max}/\text{MeV}}\right] \tag{1.0.12}$$

#* Use photoelectric shell energies / Use impact ionization shell energies

Type: logical

True or false includes or neglects photoelectric shell energies/impact ionization shell energies in the photon/electron energy grouping respectively. These energies depend on the materials specified by the user.

### Physics

#* Electron inelastic scattering cross section

Type: character
Options: 2

- `Moller`: Uses Möller scattering as the inelastic scattering cross section [3]. Involves no treatment of binding energies.
- `RBED`: Uses the relativistic binary encounter dipole (RBED) cross section as the inelastic scattering cross section [2]. Notably, this may incur a small, once-per-run slowdown that is currently being optimized. However, in general, this cross section is recommended for lower energies, as it is a bit more respectful of atomic electronic structure than Möller scattering.

See **wiscobolt physics** for a better understanding of Möller scattering and the RBED cross section.

#* Electron RCSDA cross section (MGXS only)

Type: character
Options: 2

- `first order`: Uses first order discretization of the RCSDA operator [3].
- `second order`: Uses second order discretization of the RCSDA operator [3].

#* Electron elastic scattering cross section

Type: character
Options: 3

- `ELSEPA`: Uses elastic scattering cross sections generated by the ELSEPA program. This program was created by Francesc Salvat, Aleksander Jablonski, and Cedric J. Powell. It is described briefly in **wiscobolt physics**, and there are documents relating to its construction here [4]. Notably, ELSEPA is recommended in general, and strongly recommended for low energy transport (sub-keV).
- `Wentzel-Moliere`: Uses the Rutherford nuclear elastic scattering cross section with the Wentzel-Moliére screening factor. Not recommended for low energies (sub-keV).
- `fitted`: Uses the fitting scheme and parameters described by Boschini et al. [5]. Not allowed at all for low energies (sub-keV).

#* Electron RCSDA cutoff (FEXS only)

Type: real
Only meaningful for FEXS. This describes the restricted continuous slowing down approximation's energy transfer cutoff Δ (in MeV), below which inelastic scattering is considered 'soft,' resulting in no angular change and a vastly different treatment for energy loss. For MGXS, this parameter is meaningless. A recommended value is 0.02 (MeV).

### Materials

# Material atomic numbers

Type: integer [user defined length] [user defined number of rows]
Material specification deserves a thorough discussion. Under this category, the user must specify the atomic composition of whatever materials they are interested in using. The atoms are listed sequentially, with one column corresponding to the atomic number (use tab as separation), and with one row corresponding to one material (in order, according to the physical tags assigned in Gmsh). At the end of each row, the user must place the integer -1. After the last material, the user must place the integer -1 in the following row.

For example, say that my problem involves water, gold, and steel. I will assume that steel is composed only of iron and carbon. My input would be:

```
 1    8   -1
79   -1
 6   26   -1
-1
```

Note that the atoms do not need to be ordered particularly, but the materials are of course ordered according to their physical tags in Gmsh.

Valid atomic numbers are $1 - 99$. Atomic number 0 is used to specify vacuum, or, cross sections that are fully zero. In the future, it is planned that numbers above 99 will be dedicated to ‘special’ materials, such as water, plastic, air, with particularly-constructed cross sections.

# Corresponding number of atoms

Type: integer [user defined length] [user defined number of rows]

Now the user must specify, for each column and row of their entry above, the number (or relative number) of atoms. The user no longer must use -1 as a delimiter for the rows. For the example above (water, gold, steel), I may use:

$$\begin{matrix} 2 & 1 \\ 1 & \\ 1 & 99 \end{matrix}$$

Importantly, for a mixture like steel, we have listed the composition by number (not mass). Here we're assuming, as an example, 1 carbon atom for every 99 iron atoms.

# Material densities

Type: integer [user defined number of rows]

Finally, the user specifies the mass density of each material provided, in g/cm^3. There is one entry per row, with no delimiters. Following the example:

$$\begin{matrix} 0.9998395 \\ 19.3 \\ 8.0 \end{matrix}$$

These inputs will successfully specify to wiscobolt a problem involving water, gold, and steel, at around room temperature and 1 atm of pressure.

### Beam definition

# Beam energy distribution

Type: character

Options: 3

- `polychromatic`: Uses a polychromatic beam spectrum that is user-defined. Specifically, the user may define beam energy spectra using a text file with the first column corresponding to energy values and the second column corresponding to (relative) number of particles. This spectrum is normalized by wiscobolt.

  `#--> Linac`: **Type: character**. The aforementioned text file must be placed in the `Linacs` folder in the main directory and the name of this text file specified under this subheader.

- `boxcar`: Uses a 'boxcar' beam spectrum. That is, it is zero outside of $E_{\max} - \Delta E \leq E \leq E_{\max}$, and a constant $1/\Delta E$ within this range.

  `#--> Energy width`: **Type: real**. The quantity ΔE is specified here. Obviously, can not exceed $E_{\max} - E_{\min}$.

- `monochromatic`: Also technically uses a boxcar beam spectrum, but makes its width exactly equal to $1/(E_1 - E_2)$, for which it is recommended to be using linear energy groups. Note that, even with a precisely chosen ΔE, the `boxcar` option does not enforce that only group $g = 1$ is populated, but this option does just that.

# Beam origin

Type: real [3]

The spatial coordinates of the beam origin, referred to later in this document as $\mathbf{r}_0$.

# Beam axis

Type: real [3]

The coordinates of the beam axis, referred to later in this document as $\hat{\mathbf{k}}_0$ following normalization. Does not need to be normalized by the user.

The beam axis, with the beam origin and a parameter to be described later, define a 'beam coordinate system,' that is the basis of a number of subroutines across the `mesh.f08` and `sources.f08` modules. Additionally, in later versions of wiscobolt, wherein one will be able to define angular and planar distributions for their source, the beam coordinate system will be the frame in which these distributions are most conveniently defined. Specifically, the coordinate system $\mathbf{r}'$ has $\hat{\mathbf{z}}' = \hat{\mathbf{k}}_0$, and then $\hat{\mathbf{x}}'$ and $\hat{\mathbf{y}}'$ are orthogonal to this direction and one another, but are more precisely defined by an active rotation with respect to the 'lab' reference frame. It is useful to conceptualize these beam axes as originating from $\mathbf{r}_0$, specifically in the context of some of the subroutines in `mesh.f08`.

# Beam angular distribution

Type: character
Options: 2

→ `spherical`: The beam is a spherical point source at $\mathbf{r}_0$. This means that source particles are created *only* in the direction $(\mathbf{r} - \mathbf{r}_0)/|\mathbf{r} - \mathbf{r}_0|$. Note that this leads to an inverse-square law in the uncollided fluence.

→ `planar`: The beam is a planar source, whose axis is the user-specified beam axis $\hat{\mathbf{k}}_0$, and including the point $\mathbf{r}_0$. Source particles are only created with direction $\hat{\mathbf{k}}_0$, which leads to an uncollided fluence that involves no inverse square law.

At the moment, these angular distributions are constant but may be shaped by beam cutouts (described in the next section). In the future, it will be possible to allow the user to define angular distributions or planar distributions so that a particular direction/ray from the source may have more particles.

Nevertheless, angular distributions have a substantial effect on the 'scale' of the output quantities in wiscobolt, in particular how they affect the factor of the source that corresponds to the total number of particles populating the problem, which is fully un-referenced in wiscobolt. That is to say, in wiscobolt, all solutions are with divided by either the number of source particles or the emission rate of source particles, and the user must apply these factors themselves (much more on this later in the chapter).

# Beam cutout

Type: character
Options: 3

→ `rectangle`: The beam cutout is a rectangle of some shape defined along the beam axis.

`#--> Beam cutout parameters`: **Type: real** [**4**]. The first entry is the distance from the beam origin along the beam axis at which the cutout is defined. The second entry is the length of the rectangle along the beam's $x-$direction. The third entry is the length of the rectangle along the beam's $y-$direction. The fourth entry is a rotation angle about the beam axis through which the beam coordinate system should be rotated. A discussion on the beam axis is given in **Section 2.1**, and so if the user intends to use a non-square beam cutout it is recommended they understand this section.

→ `circle`: The beam cutout is a circle of some radius defined along the beam axis.

`#--> Beam cutout parameters`: **Type: real** [**2**]. The first entry is the distance from the beam origin along the beam axis at which the cutout is defined. The second entry is the radius of the beam at this distance.

→ `none`: There is no beam cutout. For planar beams, this means the user will actually have to define the cross section of the problem volume from the perspective of the planar beam (defined, of course, with the beam axis and origin), more on this later. For spherical beams, the user will either need to provide the total number of particles created by the source, or the solid angle spanned by the volume from the perspective of the beam origin (the former is much simpler), more on this later as well.

# Extra options

There are a number of extra options the user may define. We will visit each of them below. They can be used in any order, but some of them require a row directly below them with certain integers/reals.

→ `use custom physics`: (MGXS only) The user wishes to provide their own cross sections. The cross sections are provided to wiscobolt in the format `[g1,g2,l+1,m]`, where `g1` is the incoming energy group, `g2` is the outgoing energy group, `l` is the Legendre order, and `m` is the material. The convention accepted for these user-defined cross sections is different from the one described in the main **wiscobolt** document, in that the `l`'th Legendre moment is defined via Equation I.10 in [3], which is considered to be more convenient for the user if they are using CEPXS-generated cross sections. If the user is using the `inelastic ETC` and/or the `Exact RCSDA angular`, their electron cross sections must have size $L+2$, where the first $L+1$ entries already have the ETC applied and the $(L+2)$th entry consists of all forward-scattering cross sections (i.e., RCSDA and/or the $(L+1)$th moment of inelastic scattering). The files are placed in the folder `Physics data/User-defined XS`. The files must be data files named `[photon/electron]_[scattering/total/energy_dep/charge_dep].dat` for the corresponding photon and electron cross sections, or for coupled photon-electron cross sections, `photon_to_electron_scattering.dat`. All cross sections must be provided.

→ `post processing`: Post-processing entails printout/creation of the following quantities for every particle type:

- Nodes in a slice of the mesh (data file)
- Faces in a slice of the mesh (data file)
- Fluence in a slice of the mesh (data file)
- Uncollided fluence in a slice of the mesh (data file)
- Angular fluence in a slice of the mesh (data file)
- Normalized energy distribution (printed in command prompt)
- Normalized polar angle distribution (printed in command prompt)

as well as enabling the next five options in this list. Now, below the `post processing` entry, one must specify an integer from $0-3$, which carry the following meaning:

0. Do not create files that describe the fluence/angular fluence/deposition in a mesh slice for the purposes of graphing.
1. Do so for a mesh slice with constant x.
2. For a mesh slice with constant y.
3. For a mesh slice with constant z.

and in the same row, a real number, which is the constant value of $x/y/z$ at which the mesh slice should be evaluated. Notably, the mesh slice must actually exist within the mesh. It will not be created by wiscobolt, though this is technically possible and not particularly un-feasible.

→ `calculate energy deposition`: Energy deposition maps will be created and saved. Also calculates volume integrals of energy deposition ('total energy deposition'). If the user specifies a mesh slice as per the above, then wiscobolt will save the energy deposition in the mesh slice as well.

→ `calculate charge deposition`: Charge deposition will be computed as described above.

→ `calculate dose deposition`: Dose deposition will be computed as described above.

→ `calculate line distributions`: For energy deposition, dose, and charge deposition, the values of these maps along the beam axis will be printed along with the beam axis nodes. This assumes that there are connected nodes along the beam axis within the mesh.

→ `planar integrals along depth`: For energy deposition, dose, and charge deposition, wiscobolt will print area integrals vs depth along the beam axis. This does *not* require any special structure for the mesh, as wiscobolt uses a method of doing these integrals at an arbitrary depth value, across the entire cross section of the mesh at that depth value. The user must specify one integer and two reals in the next row:

- **integer**: Number of points along the depth axis at which the area integrals are evaluated.
- **real**: Smallest value of depth, with respect to the beam axis (increasing values are going *away* from the beam origin).
- **real**: Largest value of depth, with respect to the beam axis.

→ `PN to SN`: Converts all P_N quantities to S_N quantities using quadrature, prior to post-processing.

→ `scale mesh`: Applies a scale to the $x-$, $y-$, and $z-$ directions of the mesh. Provide three reals in the next row, corresponding to these scale factors. The identity is 1.0 for each of these entries.

→ `translate mesh`: Applies a translation to the $x-$, $y-$, and $z-$ directions of the mesh. Provide three reals in the next row, corresponding to a translation vector. The identity is 0.0 for each of these entries.

→ `use stored [photon/electron] solution`: wiscobolt saves the angular fluence, the fluence, and uncollided fluence of any particle solved, no matter what. There is thus the option for the user to use a stored solution, as long as it is named, formatted, and placed exactly where it is left by wiscobolt, namely, in the `Results` folder in the main directory. The discretization parameters and mesh of the stored solution must match those of the new solution.

→ `store sweep`: For S_N, one can choose to store the so-called 'sweep' arrays in the mesh folder, which makes re-using the given mesh faster by virtue of not having to create these sweep arrays (the entire `sweep_order.f08` module is foregone). The sweeps correspond to specific angular quadrature sets, so they are stored for a choice of angular discretization parameter. The option to reuse a sweep is to be discussed next. Note that sweep files can be quite large, on the order of a few GB for meshes with $100,000$ elements with angular discretization parameter of 8.

→ `use stored sweep`: For S_N, uses sweep stored in mesh file (if it exists, if it does not, the program ends). It must be noted that applying a non-uniform scale to your mesh changes the sweep order, but translating it does not, and so if you scale your mesh between two runs, you may not use a stored sweep.

1.1 Output

All outputs are stored in the `Results` folder of the main wiscobolt directory. Within this folder, a run is stored in a folder which is named by the date in the format `mm-dd-yy`, and time in the format `hh-mm-ss`, so the folder is named `mm-dd-yy_hh-mm-ss`. The command prompt printout as well as the input file that was used are always stored as well. All 2D outputs, such as energy spectra, depth integrals, etc., will be printed as separate columns for the x and $y-$axes in the command prompt, whereas more cumbersome arrays such as the fluence will be stored in .dat files. All objects are discretized for a given index set according to the discretization methods described in the main **wiscobolt** document. We will briefly summarize them here and note their correspondence to indices that are used in certain discretization methods.

→ `[e,k]`: Elemental node index. A function $f(\mathbf{r})$ is given as $f_k^e = f(\mathbf{r}_k^e)$, where $\mathbf{r}_k^e$ is the kth node in the eth element of the mesh (these are ordered according to Gmsh).

→ `[k]`: Global node index. A function $f(\mathbf{r})$ is given as $f_k = f(\mathbf{r}_k)$, where $\mathbf{r}_k$ is the kth node in the global mesh (these are ordered according to Gmsh).

→ `[i,j]`: S_N angular indices. A function $f(\hat{\mathbf{k}})$ is given as $f_{ij} = f(\hat{\mathbf{k}}_{ij})$. The precise angular ordinates $\hat{\mathbf{k}}$ are based on the quadrature rules described in the main **wiscobolt** document, that is, $\mu_i = \cos\theta_i$ is given at Gauss-Legendre quadrature abscissae of order given by the angular discretization parameter, and ϕ_j is given by Chebyshev quadrature abscissae, which have a simple analytic expression.

→ `[q]`: P_N angular indices. A function of $f(\hat{\mathbf{k}})$ is expanded with respect to real spherical harmonics described in this document in (1.0.6). The expansion coefficients are given generally as f_ℓ^m, but in wiscobolt we use the index $q(\ell, m) = \ell(\ell+1) + m + 1$, which maps to ℓ and m uniquely, with $\ell(q) = \lceil\sqrt{q}\rceil - 1$ and then $m(q) = q - \ell(\ell+1) - 1$.

→ `[g]`: Energy indices.

- MGXS: Group *average* quantities. This is in contrast to the group *total* quantities, which is what is used in the main **wiscobolt** document, and used by wiscobolt except when a quantity is output. That is, a function $f(E)$ is given as $f_g = \int_{E_{g+1}}^{E_g} dE\ f(E)/\Delta E_g$ *only* in the data file, and nowhere else in wiscobolt.
- FEXS: Global energy node quantities. A function $f(E)$ is given as $f_g = f(E_g)$, where E_g is the gth node in the global energy mesh.

→ `[n,g]`: FEXS global energy mesh indices. A function $f(E)$ is given as $f_n^g = f(E_n^g)$.

→ `[S]`: Nodes in a mesh slice. This is unique to quantities that are written over a mesh slice. The mesh slice nodes are exported as described shortly.

→ `[F]`: Faces in a mesh slice. Also unique to mesh slice quantities, and also described shortly.

Now let's discuss the format of every data file that is created.

→ Angular fluence.

- Fundamental units: MeV^{-1}-cm^{-2}-sr^{-1}
- Name: `[photon/electron]_angular_fluence.dat`. Uses `photon` or `electron` depending on which particle has been solved.
- Indices | shape:
  - S_N MGXS: `[k,i,j,g]` | `[NK,A,2*A,G]`
  - S_N FEXS: `[k,i,j,g]` | `[NK,A,2*A,G+1]`
  - P_N MGXS: `[k,q,g]` | `[NK,A**2,G]`
  - P_N FEXS: `[k,q,g]` | `[NK,A**2,G+1]`

  where `NK` is the number of nodes in your mesh, `A` is the angular discretization parameter (polar angle cosine), `2*A` is twice the angular discretization parameter (azimuth), and `G` is the number of energy groups for the particle in question.
- Description: If the FCS is on, the *collided* angular fluence of the problem. That is, the total angular fluence ψ is decomposed in wiscobolt into a sum of the collided angular fluence $\tilde{\psi}$ and the uncollided angular fluence $\tilde{\psi}^0$. Otherwise, the *total* angular fluence of the problem. The value given at a node is the average of the solution across different elements of the mesh. This results in drastically less storage as well as the ability for the user to plot the solution.

→ Fluence.

- Fundamental units: MeV^{-1}-cm^{-2}
- Name: `[photon/electron]_fluence.dat`.
- Shape:
  - `[NE,NKe,G]` (MGXS)
  - `[NE,NKe,G+1]` (FEXS)
- Description: The *total* fluence of the problem.

→ Uncollided fluence.

- Fundamental units: MeV^{-1}-cm^{-2}
- Name: `[photon/electron]_uncollided_fluence.dat`.
- Shape:

  - `[NE,NKe,G]` (MGXS)
  - `[NE,NKe,G+1]` (FEXS)
- Description: The *uncollided* fluence of the problem. That is, the uncollided angular fluence $\tilde{\psi}^0$ has been factored into $\tilde{\varphi}^0\delta^2\big[\hat{\mathbf{k}}-\hat{\mathbf{k}}_0(\mathbf{r})\big]$, where $\tilde{\varphi}^0$ is the uncollided fluence. All outputs relating to uncollided fluence are neglected if the FCS is off.

→ Energy deposition/Charge deposition/Dose deposition.

- Fundamental units: MeV-cm^{-3} / e-cm^{-3} / MeV-g^{-1}
- Name: `[energy/charge/dose]_deposition.dat`.
- Shape: `[NK]`.

→ Total energy deposition/Charge deposition/Dose deposition.

- Fundamental units: MeV / e / MeV-g^{-1}-cm^3

→ Nodes in a slice of the mesh.

- Fundamental units: cm
- Name: `slice_nodes_[XY/YZ/XZ].dat`. Uses `XY`, `YZ`, or `XZ` depending on which slice you specified in `Extra options`.
- Shape: `[2,S]`, where `S` is the number of nodes in the selected slice of your mesh. You don't necessarily need to know `S`, any program you wish to use to plot/post-process should be able to read this array and determine `S` given that the first index is known to be size 2.

→ Faces in a slice of the mesh.

- Fundamental units: none
- Name: `slice_faces_[XY/YZ/XZ].dat`.
- Shape: `[F,3]`, where `F` is the number of faces in the selected slice of your mesh.

→ Angular fluence in a slice of the mesh.

- Fundamental units: MeV^{-1}-cm^{-2}-sr^{-1}
- Name: `slice_[photon/electron]_angular_fluence.dat`
- Shape:
  - `[G,A,2*A,S]` (S_N MGXS)
  - `[G+1,A,2*A,S]` (S_N FEXS)
  - `[G,A**2,S]` (P_N MGXS)
  - `[G+1,A**2,S]` (P_N FEXS)

→ Fluence in a slice of the mesh.

- Fundamental units: MeV^{-1}-cm^{-2}

- Name: `slice_[photon/electron]_fluence.dat`
- Shape:
  - `[G,S]` (MGXS)
  - `[G+1,S]` (FEXS)

$\rightarrow$ Uncollided fluence in a slice of the mesh.

- Fundamental units: MeV^{-1}-cm^{-2}
- Name: `slice_[photon/electron]_uncollided_fluence.dat`
- Shape:
  - `[G,S]` (MGXS)
  - `[G+1,S]` (FEXS)

$\rightarrow$ Energy deposition/Charge deposition/Dose in a slice of the mesh.

- Fundamental units: MeV / e / MeV-g^{-1}
- Name: `slice_[energy/charge/dose]_deposition.dat`
- Shape: `[S]`.

The final six entries of the above list are optional, and only included if the `post processing` extra option is included.

The units of every printed quantity will be normalized to a quantity related to the total number of particles introduced to the system, which the user will need to apply manually. That is to say, wiscobolt only generates response per individual source particle. However, in the even that the user does not use a beam cutout, there is not sufficient information but to normalize to the areal/angular density. That is to say, the following units are technically appended to the 'fundamental units' of all non-mesh-related outputs that are described above:

$\rightarrow$ If using spherical beam with no cutout: (no. of particles/unit beam solid angle)$^{-1}$

$\rightarrow$ If using planar beam with no cutout: (no. of particles/unit beam area)$^{-1}$

$\rightarrow$ Otherwise: (no. of particles)$^{-1}$

If a beam cutout is provided for either the spherical beam or planar beam, wiscobolt automatically applies the corresponding 'beam solid angle' or 'beam area' factor to the solution, so the user needs only to provide the total number of particles populating the system.

2 Mesh

2.1 Beam coordinate system

The beam coordinate system could be considered to have an origin given by $\mathbf{r}_0$, provided by the user, in the program's frame of reference. Iin principle, this amounts to a translation applied to take some vector $\mathbf{r}$ from one frame to another, however this will not be considered a feature of the coordinate system and is just applied where needed (such as in determining which nodes and elements are within a beam, as well as ray-tracing, etc.). So, the beam coordinate system $(\hat{\mathbf{x}}', \hat{\mathbf{y}}', \hat{\mathbf{z}}')$ is defined as follows.

There are two degrees of freedom in the transformation to the beam coordinate system. wiscobolt constructs them both from the 'beam axis,' or:

$$\hat{\mathbf{z}}' = \hat{\mathbf{k}}_0 \tag{2.1.1}$$

where $\hat{\mathbf{k}}_0$ is provided by the user as described previously (if they provide one that is not normalized, it is normalized by wiscobolt). From the user's input, we extract polar coordinates corresponding to this vector:

$$\begin{aligned} \theta_0 &= \mathrm{acos}(k_{0,z}) \\ \phi_0 &= \mathrm{atan2}(k_{0,y}, k_{0,x}) \end{aligned} \tag{2.1.2}$$

From these, a mutually orthogonal coordinate system for the beam is constructed, albeit while taking the $\hat{\mathbf{x}}'$ and $\hat{\mathbf{y}}'$ basis vectors as the vectors you would obtain if you applied a rotation to take the 'lab' frame of reference's $\hat{\mathbf{z}}$ to $\hat{\mathbf{k}}_0$ (the lab frame of reference is essentially the frame of reference of the mesh). The transformation is therefore given by:

$$\begin{aligned} \hat{\mathbf{x}}'(\theta_0, \phi_0) &= \cos\phi_0 \cos\theta_0 \hat{\mathbf{x}} + \sin\phi_0 \cos\theta_0 \hat{\mathbf{y}} - \sin\theta_0 \hat{\mathbf{z}} \\ \hat{\mathbf{y}}'(\theta_0, \phi_0) &= -\sin\phi_0 \hat{\mathbf{x}} + \cos\phi_0 \hat{\mathbf{y}} \end{aligned} \tag{2.1.3}$$

which happen to be identical to the expressions for the unit vectors commonly denoted $\hat{\mathbf{e}}_\theta$ and $\hat{\mathbf{e}}_\phi$, respectively, in the spherical coordinate system. Now, because square beam cutouts are not symmetric, the user can specify an angle δ_0 (which is the fourth or third entry of the `#--> Beam cutout parameters` subheader, depending on the cutout type) by which the beam is further rotated counter-clockwise about its axis. Thus, we could say:

$$\begin{aligned} \hat{\mathbf{x}}'' &= \hat{\mathbf{x}}' \cos\delta_0 - \hat{\mathbf{y}}' \sin\delta_0 \\ \hat{\mathbf{y}}'' &= \hat{\mathbf{x}}' \sin\delta_0 + \hat{\mathbf{y}}' \cos\delta_0 \end{aligned} \tag{2.1.4}$$

If we now take $\hat{\mathbf{x}}' \leftarrow \hat{\mathbf{x}}''$ and $\hat{\mathbf{y}}' \leftarrow \hat{\mathbf{y}}''$, we can write:

$$\boxed{\begin{aligned} &R \equiv \begin{pmatrix} \cos\phi_0 \cos\theta_0 \cos\delta_0 + \sin\phi_0 \sin\delta_0 & \sin\phi_0 \cos\theta_0 \cos\delta_0 - \cos\phi_0 \sin\delta_0 & -\sin\theta_0 \cos\delta_0 \\ \cos\phi_0 \cos\theta_0 \sin\delta_0 - \sin\phi_0 \cos\delta_0 & \sin\phi_0 \cos\theta_0 \sin\delta_0 + \cos\phi_0 \cos\delta_0 & -\sin\theta_0 \sin\delta_0 \\ k_{0,x} & k_{0,y} & k_{0,z} \end{pmatrix} \\ &\begin{pmatrix} \hat{\mathbf{x}}' \\ \hat{\mathbf{y}}' \\ \hat{\mathbf{z}}' \end{pmatrix} = R \begin{pmatrix} \hat{\mathbf{x}} \\ \hat{\mathbf{y}} \\ \hat{\mathbf{z}} \end{pmatrix} \end{aligned}} \tag{2.1.5}$$

For the user, it is important to understand the beam orientation for their problem, especially how it relates to cutouts and beam anisotropy (a WIP feature). For implementation, the beam coordinate system is not as important. The modular structure of wiscobolt is such that user beam specification actually only comes into play in two tasks; determination of the nodes in a beam (which can be foregone depending on the user's provided mesh file), and ray-tracing. In both of these tasks, invariant quantities such as dot-products are made the primary objects of interest, so the use of different coordinate systems is not very important.

References

[1] T. Wareing, J. Morel, and D. Parsons, *A first collision source method for ATTILA, an unstructured tetrahedral mesh discrete ordinates code*, 1998.

[2] C. Drumm and R. P. Kensek, *Scattering cross sections for electron transport using energy-finite-element weighting*, 2017.

[3] L. Lorence, J. Morel, and G. Valdez, *Physics guide to CEPXS: a multigroup coupled electron-photon cross-section generating code*, tech. rep. (Sandia National Labs, 1989).

[4] F. Salvat, A. Jablonski, and C. J. Powell, "ELSEPA - dirac partial-wave calculation of elastic scattering of electrons and positrons by atoms, positive ions, and molecules", Computer Physics Communications (2005).

[5] M. Boschini, C. Consolandi, M. Gervasi, S. Giani, D. Grandi, V. Ivanchenko, P. Nieminem, S. Pensotti, P. Rancoita, and M. Tacconi, "An expression for the mott cross section of electrons and positrons on nuclei with z up to 118", Radiation Physics and Chemistry (2013).