Bmad Converter Model

John Mastroberti

June 13, 2020

Contents

1 Introduction

This monograph discusses the converter model used in converter lattice elements in *Bmad*. In a converter, incoming particles generate particles of a different type. For example, a converter may be used to simulate the production of positrons produced when a tungsten target plate is bombarded by electrons.

The converter model currently in *Bmad* discussed here replaces an older model that was developed by Daniel Fromowitz[**b:fromowitz**]. This older model used equations to model the output distribution. The problems with this approach were the approximations that were used in developing the equations coupled with uncertainty as to how to calculate the various coefficients that were needed if parameters like the target material or the species of particles simulated were varied.

The present model replaces the equations of the older model with probability distribution tables for the energy and radial distribution of the outgoing particle along with a generalized parameterization of the probability distribution of the outgoing particles's direction of propagation. Probability distribution table values and coefficients needed to characterize the velocity distribution are obtained through a Geant[**geant**] simulation. These tables and coefficients can then be stored in a *Bmad* lattice file and used for efficient generation of outgoing particles. The present model is not only more accurate but can also simulate a wider range of parameters in terms of converter thickness, converter material, incoming particle energy, and different particle species.

While it would be technically feasible to use Geant directly to simulate the converter process, the production of outgoing particles in the converter is a stochastic process, the details of which are computationally expensive. The use of probability distribution tables, which only have to be computed once, while not as accurate, speeds up the computation time by orders of magnitude.

The imputus for developing the new model was to better simulate the converter in the Cornell CESR Linac which generated positrons due to bombardment of a tungsten plate by electrons. The electrons have an energy of order $\sim 100\,\text{MeV}$. As the incident electrons pass through the converter, they emit photons via Bremsstrahlung, which in turn decay to e^+e^- pairs:

$$e^- + Z \to e^- + Z + \gamma \to e^- + Z + e^+ + e^- \tag{1}$$

2 The converter model

The probability distribution is computed assuming assuming that the incoming particle's momentum is perpendicular to the surface of the converter. This is reasonable since deviations of the probability distribution will be second order in the transverse momentum. The coordinate system is shown in Figure 1. The outgoing particle's position on the downstream face of the converter is then described by r, its distance from the z axis, and the angle θ shown in the figure. By symmetry, θ must be distributed uniformly between 0 and 2π and so the probability distributions will be independent of θ. The x-axis is defined to be in the same direction as $\mathbf{r}$. We then choose our y axis such that (x, y, z) is a right handed orthogonal coordinate system.

2.1 Probability Distributions

The incoming particle is labeled by a plus symbol subscript and the outgoing particle is labeled by a minus symbol subscript. The outgoing particle as it leaves the surface of the converter is characterized by θ, its momentum p_+, the offset from the origin r, and the dx/ds and dy/ds slopes with

$$p_+c = |\mathbf{p}_+|\, c \tag{2}$$

$$\frac{dx}{ds} = \frac{p_x}{p_s} \tag{3}$$

$$\frac{dy}{ds} = \frac{p_y}{p_s} \tag{4}$$

Ignoring θ, we seek the distribution

$$P\left(p_+c, r, \frac{dx}{ds}, \frac{dy}{ds}\right) \tag{5}$$

which describes the probability that an outgoing particle will attain particular values of p_+c, r, $\frac{dx}{ds}$, and $\frac{dy}{ds}$. This probability distribution is dependent upon the incoming particle's energy, as well as the thickness and material type of the converter. This will be discussed later.

P is normalized to the number of outgoing particles produced per incoming particle:

$$\int P\left(p_+c, r, \frac{dx}{ds}, \frac{dy}{ds}\right) d(p_+c)\, dr\, d\left(\frac{dx}{ds}\right) d\left(\frac{dy}{ds}\right) = \frac{N_+}{N_-} \tag{6}$$

This normalization lets us easily account for the fact that number of outgoing particles produced varies with the incoming particle energy and converter thickness. P can be decomposed into two distributions:

$$P\left(p_+c, r, \frac{dx}{ds}, \frac{dy}{ds}\right) = P_1\left(p_+c, r\right) P_2\left(\frac{dx}{ds}, \frac{dy}{ds}; p_+c, r\right), \tag{7}$$

where we choose P_1 to be normalized to N_+/N_- and P_2 to be normalized to 1.

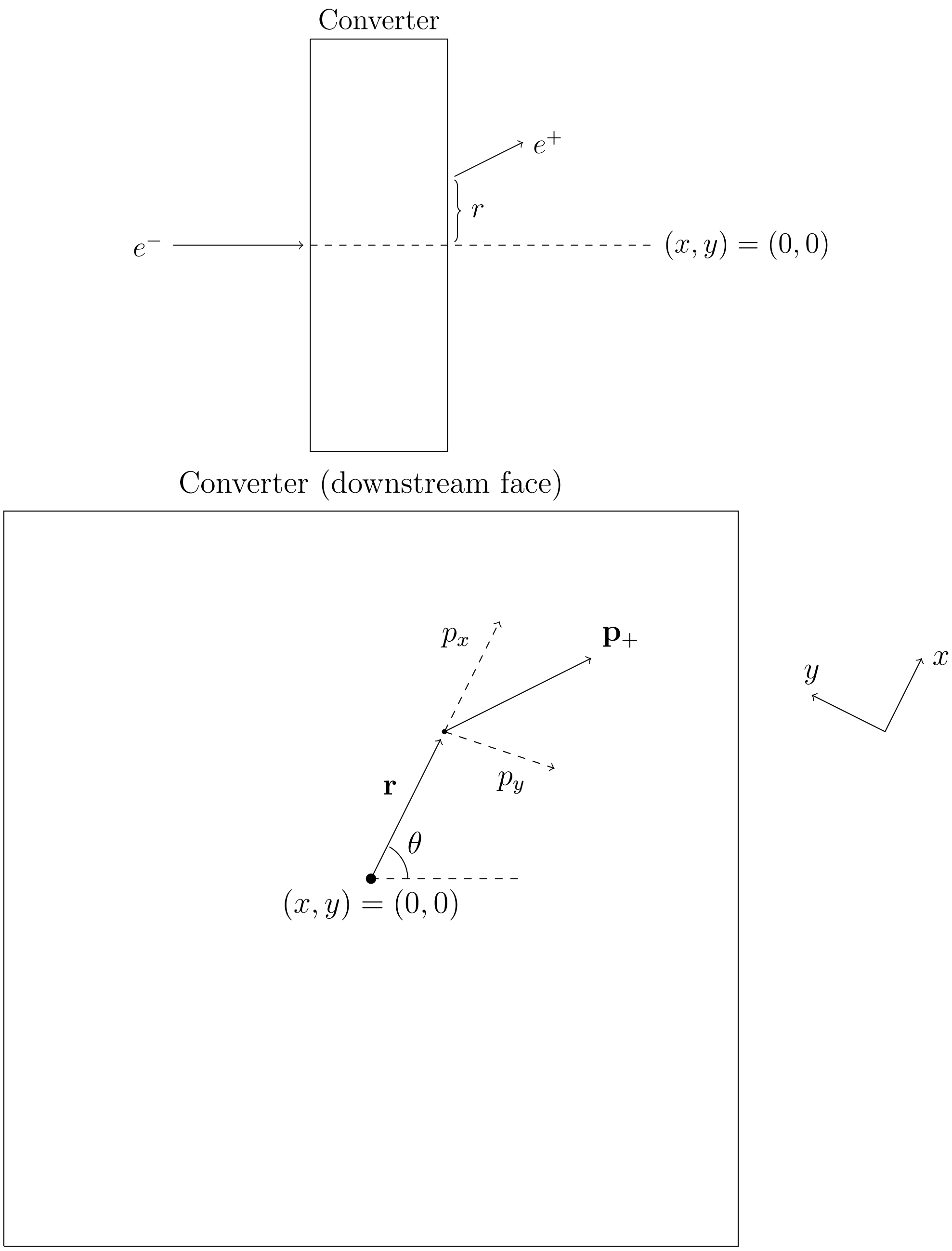

Figure 1: Coordinates used to describe the outgoing particles exiting the converter. The incoming particle is labeled e^- and the outging particle is labeled e^+.

2.2 P_1 and P_2 Parameterization

Using Geant[**geant**], A number of incoming particles of a given energy incident upon a converter of a given thickness is simulated. The values of p_+c, r, $\frac{dx}{ds}$, and $\frac{dy}{ds}$ for each outgoing particle at the downstream face of the converter is recorded. This data is then binned into a two-dimensional histogram by p_+c and r. The sizes of the bins are chosen non-uniformly so that each bin holds approximately the same number of outgoing particles. The binned data produces a probability distribution table that characterizes $P_1(p_+c, r)$.

To model P_2, for each (p_+c, r) bin, the distribution of particles in dx/ds and dy/ds is fit to the functional form

$$P_2\left(\frac{dx}{ds}, \frac{dy}{ds}; p_+c, r\right) = A\frac{1+\beta\frac{dx}{ds}}{1+\alpha_x^2\left(\frac{dx}{ds}-c_x\right)^2+\alpha_y^2\left(\frac{dy}{ds}\right)^2}. \tag{8}$$

Since P_2 is normalized to 1, A is not a true fit parameter, but is fixed by the normalization. The fit gives values of c_x, α_x, α_y, and β in each (p_+c, r) bin. The distribution of $\frac{dx}{ds}$ and $\frac{dy}{ds}$ is also characterized by $\frac{dx}{ds}_{min}$, $\frac{dx}{ds}_{max}$ and $\left|\frac{dy}{ds}\right|_{max}$, which define the rectangle in $\frac{dx}{ds} \times \frac{dy}{ds}$ space where $P_2\left(\frac{dx}{ds}, \frac{dy}{ds}\right)$ is significantly nonzero. Fits to each of the parameters $c_x, \alpha_x, \alpha_y, \beta, \frac{dx}{ds}_{min}, \frac{dx}{ds}_{max}, \left|\frac{dy}{ds}_{max}\right|$ as functions of p_+c and r are made as follows:

- At each value of p_+c below a user-defined cutoff, we perform a 1D fit of the form

  $$\pi(r) = a_0 + a_1 r + a_2 r^2 + a_3 r^3 + a_4 r^4 \tag{9}$$

  for each parameter $\pi = c_x, \alpha_x, \alpha_y, \beta, \frac{dx}{ds}_{\text{min}}, \frac{dx}{ds}_{\text{max}}, \left|\frac{dy}{ds}\right|_{\text{max}}$. For c_x and β, a_0 is fixed to be 0, as c_x and β must be zero at $r = 0$ by symmetry. For all other parameters, a_4 is fixed to be zero, so that a third degree polynomial is fit instead of a fourth degree polynomial.

- Above the user-defined p_+c cutoff, we perform a 2D fit of the form

  $$\pi(r) = (1 + a_1(p_+c) + a_2(p_+c)^2 + a_3(p_+c)^3)(b_0 + b_1 r + b_2 r^2 + b_3 r^3)e^{-(k_p(p_+c)+k_r r)} + C \tag{10}$$

  for each parameter $\pi = c_x, \alpha_x, \alpha_y, \beta, \frac{dx}{ds}_{\text{min}}, \frac{dx}{ds}_{\text{max}}, \left|\frac{dy}{ds}\right|_{\text{max}}$. The parameter C is only used for $\frac{dx}{ds}_{\text{max}}$. Note that we set the constant term for the p_+c polynomial to 1 in each case. This must be done so that the fitting problem is full rank.

With these fits in hand, we have an approximation of $P_2\left(\frac{dx}{ds}, \frac{dy}{ds}; p_+c, r\right)$.

3 Setup for Generating the Probability Parameters

Generating the probability parameters has two main stages. In the first stage, the particle creation events are simulated with Geant, and the resulting outgoing particles are binned into a histogram. In the second stage, fits are performed for $\frac{dx}{ds}$ and $\frac{dy}{ds}$. Each stage has an associated executable: `converter_simulation` for the first stage, and `converter_fitter` for the second.

After the probability parameters are generated, a *Bmad* lattice file can be created that contains these parameters and this lattice file is used for simulating the converter independent of Geant.

3.1 Dependencies

Builds require a C++ compiler with support for C++17. GCC 8 or higher should be fine.

A built *Bmad* Distribution or Release is a prerequisite. See the *Bmad* web pages (or your local *Bmad* Guru) for details.

Needed is an up to date installation of Geant4 on your system. See the Geant4 installation guide below for details on how to get Geant4 up and running on Linux. You will also need `cmake` version 3.8 or greater installed.

3.2 Geant4 Installation Guide

This guide is an abbreviated version of the instructions found on the Geant4 website.

1. Create a directory, here called `GEANT_DIR`, where Geant will be installed and `cd` to this directory.

2. Download the .tar.gz source files archive from https://geant4.web.cern.ch/support/download into `GEANT_DIR` and unpack with
```
tar xzvf geant4.10.06.tar.gz
```
   Change the version number as appropriate. There should now be a directory `geant4.10.06`.

3. Make a sub-directory where you will build Geant with
```
mkdir geant-build
```

4. `cd` to this new directory, and use `cmake` to configure the Geant4 build with
```
cmake -DGEANT4_INSTALL_DATA=ON \
      -DCMAKE_INSTALL_PREFIX=$GEANT_DIR/geant4-build \
      ../geant4.10.06
```
   Change the version number as appropriate. The first `-D` flag will cause the necessary data sets to be downloaded when we build Geant, and the second `-D` flag sets the install directory.

   Note: if you encounter the the error

```
Could NOT find EXPAT (missing: EXPAT_LIBRARY EXPAT_INCLUDE_DIR)
```

at this step, try editting the file

```
../geant4.10.06/cmake/Modules/Geant4OptionalComponentents.cmake
```

replacing the line

```
option(GEANT4_USE_SYSTEM_EXPAT "Use system Expat library" ON)
```

with

```
option(GEANT4_USE_SYSTEM_EXPAT "Use system Expat library" OFF)
```

and then re-run the above `cmake` command.

5. After `cmake` finished running, start building Geant with

   ```
   make -j4
   ```

   This will run four threads. You can change the number of threads to increase or decrease the compile speed.

6. Once the compilation has finished, install with

   ```
   make install
   ```

3.3 Compiling the Executables

The converter simulation and fitting programs are distributed as part of any *Bmad* Distribution or Release and are located in the `util_programs` directory. [If you are not sure where your Distribution or Release is, ask you local *Bmad* Guru.] You may need to make a local copy of the `util_programs` directory if you do not have write access to the Distribution or Release version. This can be done with the command:

```
$ svn co https://accserv.lepp.cornell.edu/svn/trunk/src/util_programs
```

The file `$GEANT_DIR/geant-build/geant4make.sh` must be sourced to add Geant4 to your path. To do so, use the following commands:

```
cd $GEANT_DIR/geant-build && source geant4make.sh
```

Also execute the command

```
export ACC_BUILD_TEST_EXES="Y"
```

Now `cd` to the `util_programs` folder, and then simply run `mk` to build `converter_simulation` and `converter_fitter`. If you get errors like:

```
util_programs/converter_element_modeling/fitter/cauchy.cpp:35:10:
error: expected unqualified-id before '[' token
    auto [xval, yval, binval] = bins[i];
         ^
```

The problem is that the C++ compiler does not support C++17.

4 How to run the programs

4.1 Configuration

Both `converter_simulation` and `converter_fitter` are configured by editing the file `config.txt`, which should be in the working directory where you run both executables. Each line in this file should have the form

```
setting = value
```

Comments can be inserted with an exclamation mark `!` and last until the end of the line. An example config file, with all available settings listed, is shown below.

```
! Example configuration file
! The ! introduces a comment that lasts until the end of the line
target_material = tungsten ! Defines the converter material
target_thickness = 6.35 mm, 1.0 cm ! Defines the target thicknesses to be simulated
pc_in = 300 MeV, 500 MeV, 1 GeV ! Defines the incoming particle energies to be simulat
out_pc_min = 0 ! Minimum pc cutoff for outgoing particles, defaults to 0
out_pc_max = 100000000 ! Maximum pc cutoff for outgoing particles (in eV here)
dxy_ds_max = 10 ! Maximum cutoff for the magnitude of dx/ds
                ! and dy/ds allowed for outgoing particles
output_directory = sim_data ! Name of the directory where data will be output,
                            ! should be specified relative to the working directory
                            ! Defaults to sim_data
num_bins = 15 ! Number of bins to use for both pc and r histogram binning
              ! with just this line, you would have a 15x15 histogram
              ! Defaults to 15
num_pc_bins = 12 ! Number of pc bins to use for histogram binning
num_r_bins = 20 ! Number of r bins to use for histogram binning
fit_crossover = 10 MeV ! For alpha and beta fits, this defines the
                       ! point where the fitter transitions from 1D
                       ! to 2D fits, defaults to 10 MeV
```

All settings accept a single value, except for `pc_in` and `target_thickness`, which accept a comma separated list of values. The settings `out_pc_min`, `output_directory`, `num_bins`, and `fit_crossover` have default values, while the settings `target_material`, `target_thickness`, `pc_in`, `out_pc_max`, and `dxy_ds_max` must be specified in the file.

The settings `pc_in`, `out_pc_min`, and `out_pc_max` take values with dimensions of energy. These default to eV if no unit is specified, although `MeV` and `GeV` can be added as suffixes to use MeV and GeV instead as shown in the sample file. The `target_thickness` setting takes values with dimensions of length. The default unit is meters, although `cm` and `mm` are supported as well.

The settings `num_bins`, `num_pc_bins`, and `num_r_bins` control the number of bins used in the histogram. If you only provide a value for `num_bins`, it will be used for both the number of p_+c bins and then number of r bins. If you provide `num_pc_bins` or `num_r_bins` in your config file, this value will supersede num_bins for the number of p_+c or r bins respectively. If you do not set num_bins in your config file, you must set both num_pc_bins and num_r_bins.

4.2 The Simulation Program

To run converter_simulation and perform the converter simulation, first create and edit the configuration file config.txt, and place it in your working directory. Then, just run

```
$ converter_simulation
```

at your command prompt. The program will parse your config file and report the settings it read, and report if there are any problems reading your config file. It will then verify that the directory you set for output_directory does not exist or is empty, and will ask you if you want to overwrite it if it already exists. Then, for each value of pc_in and target_thickness specified in the config file, the program will simulate many particle creation events for those settings. For example, with the above config file, six simulations will be run with the following settings:

- 300 MeV pc_in and 6.35 mm target_thickness
- 500 MeV pc_in and 6.35 mm target_thickness
- 1 GeV pc_in and 6.35 mm target_thickness
- 300 MeV pc_in and 1 cm target_thickness
- 500 MeV pc_in and 1 cm target_thickness
- 1 GeV pc_in and 1 cm target_thickness

Depending on your computer and the number of different simulations that need to be run, this step may take several hours.

4.3 The Fitting Program

Once the simulations are complete, just run

```
converter_fitter
```

in the same directory where you ran converter_simulation. converter_fitter will reparse your config file for the settings it needs, and will again report on any errors it encounters. It then performs the fit from Equation 8 in each of the (p_+c, r) bins for each simulation. At this stage, the program may report that the fitting iteration limit has been reached a few times; this is not cause for concern. Once this step is complete, and the program has obtained values of c_x, α_x, α_y, and β in each (p_+c, r) bin, it performs the fits from Equations 9-10 on these fit parameters. Finally, the results of the simulation, as well as the results of the fits, are output to the file converter.bmad, located in the output_directory specified in the config file.

5 Output from the Programs

5.1 Simulation Output

After running the `converter_simulation` program, the directory specified by the `output_directory` setting in the configuration file will exist in your working directory. Inside it, there will be one file of the format '`E{pc_in}_T{thickness}_er.dat`', for each incoming p_c and target thickness specified in the configuration file, where `pc_in` and `thickness` are the p_c and thickness in MeV and cm respectively. These files contained the binned data which approximate $P_1(p_+c, r)$. The output directory will also contain a directory `dir_dat`, with subdirectories `E{pc_in}_T{thickness}_er.dat` for each p_c and target thickness combination. Each of these directories will contain files named `E{pc_out}_r{r_out}_bin.dat`, which contain the binned $\frac{dx}{ds}$ and $\frac{dy}{ds}$ data used by `converter_fitter`.

5.2 Fitting Output

After running the `converter_fitter` program, the output directory will also contain a file called `converter.bmad`. This file aggregates all the information about P_1 and P_2 at each p_+c and target thickness tested, and is designed for use with a *Bmad* converter element.

5.2.1 Gnuplot Files

`converter_fitter` also generates several gnuplot scripts for inspecting the quality of the obtained fits. These are all written to the individual `E{}_r{}` directories under `dir_dat`.

Each of the (p_+c, r) bins gets two gnuplot scripts: `cauchy_E{pc_out}_r{r_out}.gp` and `meta_E{pc_out}_r{r_out}.gp`. The scripts with the `cauchy` prefix display the distribution P_2 obtained by directly fitting Equation 8 to the data in each bin. The scripts with the `meta` prefix display the distribution P_2 obtained from evaluating the fits from Equations 9-10 to the Cauchy fit parameters.

`converter_fitter` also outputs scripts for viewing the fits from Equations 9-10 across all (p_+c, r) bins. These are named `c_x_master.gp`, `a_x_master.gp`, `a_y_master.gp`, `beta_master.gp`, `dxds_min_master.gp`, `dxds_max_master.gp`, and `dyds_max_master.gp`.

To view any of these plots, simply open Gnuplot in the `E{}_T{}` directory of interest, and call the script. For example:

```
> cd /home/user/sim_data/dir_dat/E300_T0.635
> gnuplot
gnuplot> call 'c_x_master.gp'
```

This will open the plot for c_x across all (p_+c, r) bins for $p_+c = 300$ MeV, $T = 0.635$ cm.

6 The *Bmad* Converter Element

As mention in the previous section, `converter_fitter` outputs a file, `converter.bmad`, which encodes all of the simulation and fitting output, and can be used to specify the properties of a *Bmad* converter element. See the *Bmad* manual for the full details regarding the converter element in *Bmad*.

The user should be aware that *Bmad*'s method of generating outgoing particles may not be completely faithful to the simulation results. In particular, *Bmad* uses linear interpolation for the $P_1(p_+c, r)$ distribution. This can cause noticeable discrepancies on the edges of the distribution, especially in the bins with the lowest values of p_+c. Since $P_1(p_+c, r)$ changes rapidly at low p_+c, and *Bmad* additionally does not generate outgoing particles with p_+c lower than the lowest value of p_+c for the bins, *Bmad* does not generate as many outgoing particles at low p_+c as it should.

A Notes for ACC Computer Users

As detailed in Section 3.1, building `converter_simulation` and `converter_fitter` requires GSL, Geant, and a C++ compiler with support for C++17. GSL is already available on the lab machines, and a build of Geant is provided at `/nfs/acc/temp/jmm699/geant`. To get access to this Geant build, you can simply add the following to your `.bashrc`:

```
cd /nfs/acc/temp/jmm699/geant/geant4.10.06.p01-build && source geant4make.sh && cd -
```

As for the C++ compiler, you can get access to GCC 8.3 by adding

```
source /opt/rh/devtoolset-8/enable
```

to your `.bashrc`.