Positron Converter Model

John Mastroberti

April 30, 2020

Contents

1 Introduction

Every electron-positron collider needs a source of positrons. At CESR, these positrons are generated in a linear accelerator using a positron converter. Until now, we have had limited ability to optimize the design of the linac, as there was no way to model the positron converter in *Bmad*. This project rectifies this problem by providing the information needed for a converter element in *Bmad*. A converter element has been developed in conjunction with this project, which should enable optimizations of the linac design.

2 Background/physics

The positron converter provides CESR with its positrons. The converter is a slab of heavy metal (usually tungsten), which is bombarded with electrons whose energy is on the order of $\sim 100\,\mathrm{MeV}$. As the incident electrons pass through the converter, they emit photons via Bremsstrahlung, which in turn decay to e^+e^- pairs:

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

3 The converter model

The production of positrons in the converter is a stochastic process, the details of which are computationally expensive to simulate. It is therefore desirable to have a model for the properties of the produced positrons in terms of probability distributions.

3.1 Coordinates

The kinematics of a positron produced in the converter are completely described by the following quantities:

- The position at which it emerges on the downstream face of the converter.
- Its momentum upon emerging from the converter.

With this in mind, we adopt the coordinate system shown in Figure 1. We take the z axis to be the direction of the incoming electron's momentum perpendicular to the surface of the converter. The outgoing positron'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. We can then largely ignore this degree of freedom from our model if we define our x axis 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.

3.2 Probability Distributions

With this coordinate system, the position of an outgoing positron is described entirely by r (and θ, chosen uniformly at random). We choose the following three parameters to charac-

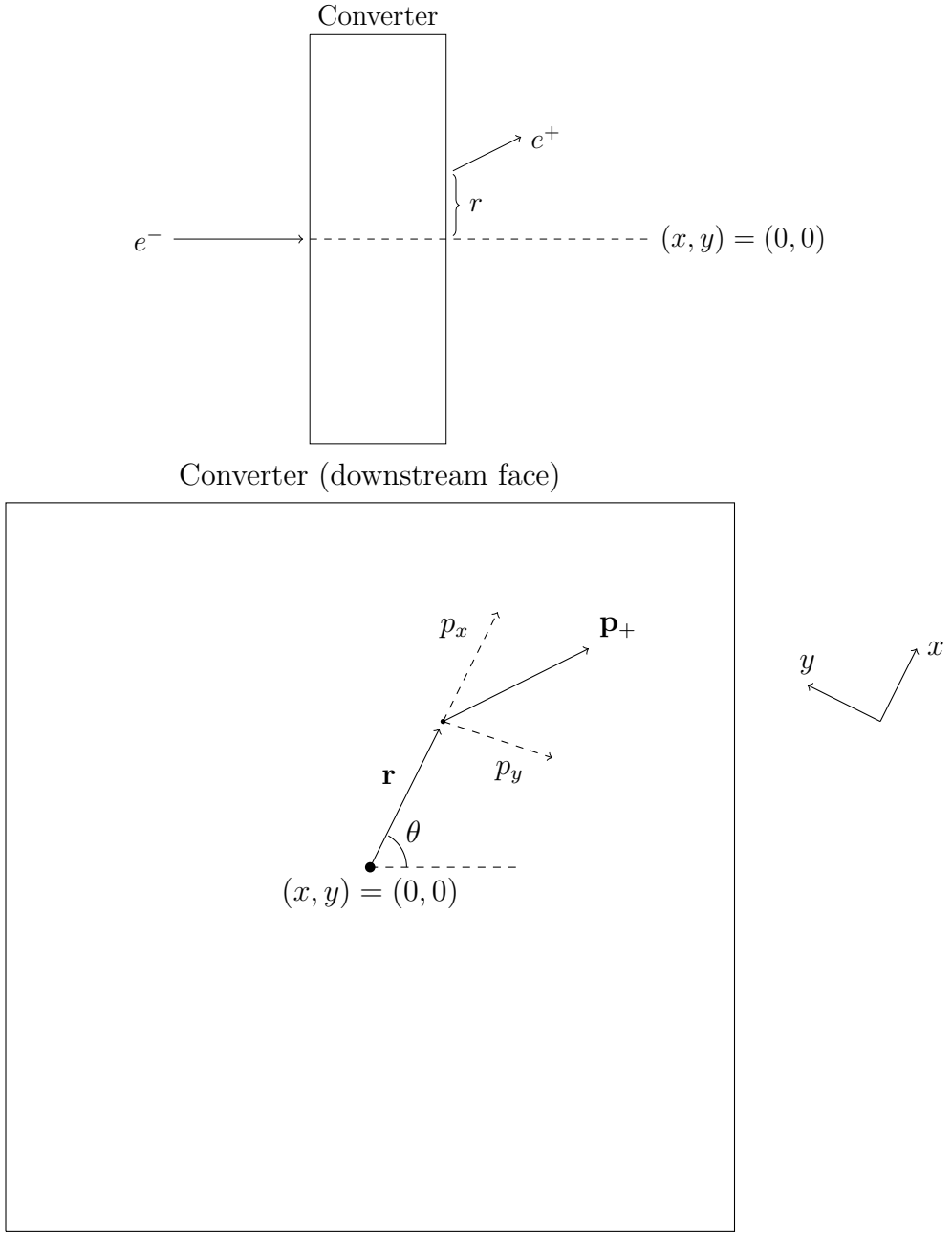

Figure 1: Coordinates used to describe the positrons exiting the converter.

terize the outgoing momentum:

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

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

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

Using these four parameters to describe outgoing positrons, 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 positron will attain particular values of p_+c, r, $\frac{dx}{ds}$, and $\frac{dy}{ds}$. This is not a true probability distribution, as it is not normalized to 1. Rather, P is normalized to the number of positrons produced per incoming electron:

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

This normalization lets us easily account for the fact that number of positrons produced varies with the incoming electron energy and converter thickness. To make the problem easier to work with, and to aid in visualization, we can separate P 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. Note that the behavior of the converter depends on the energy of the incoming electrons and the converter's thickness, so we will have a different P (and P_1 and P_2) at each incoming electron energy and target thickness.

3.3 Obtaining P_1 and P_2

Using Geant[**geant**], we simulate many incoming electrons of a given energy incident upon a converter of a given thickness. We then record the values of p_+c, r, $\frac{dx}{ds}$, and $\frac{dy}{ds}$ for each outgoing positron at the downstream face of the converter. 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 positrons. This histogram is recorded and gives us a discrete sampling of $P_1(p_+c, r)$.

Then, in each (p_+c, r) bin, we fit the functional form

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

Since P_2 is normalized to 1, A is not a true fit parameter, but is fixed by normalization. This gives us values of c_x, α_x, α_y, and β in each (p_+c, r) bin. We then perform fits to each of these parameters as functions of p_+c and r as follows:

- For c_x, we fit

$$c_x(p_+c, r) = (a_0 + a_1(p_+c) + a_2(p_+c)^2 + (p_+c)^3)(b_0 + b_1 r + b_2 r^2 + b_3 r^3) \tag{9}$$
$$\tag{10}$$

  at every value of p_+c and r.

- For α_x and α_y, we perform a series of 1D fits

$$\alpha_{x,y}(r) = (a_0 + a_1 r + a_2 r^2 + a_3 r^3)e^{-kr} \tag{11}$$

  at each value of p_+c for low p_+c. At high values of p_+c, we fit

$$\alpha_{x,y}(r) = (a_0 + a_1(p_+c) + a_2(p_+c)^2 + (p_+c)^3)(b_0 + b_1 r + b_2 r^2 + b_3 r^3)e^{-(k_p(p_+c)+k_r r)} \tag{12}$$
$$\tag{13}$$

- For β, we perform a series of 1D fits

$$\beta(r) = a_0 + a_1 r + a_2 r^2 + a_3 r^3 + a_4 r^4 \tag{14}$$

  at each value of p_+c for low p_+c. At high values of p_+c, we fit

$$\beta(p_+c, r) = (a_0 + a_1(p_+c) + a_2(p_+c)^2 + (p_+c)^3)(b_0 + b_1 r + b_2 r^2 + b_3 r^3) \tag{15}$$
$$\tag{16}$$

Note that for each of the two-dimensional polynomials, we set the coefficient on the highest power of p_+c to be 1. This must be done so that the fitting problem is full rank. With these fits in hand, we have an approximation of $P_2\left(\frac{dx}{ds}, \frac{dy}{ds}; p_+c, r\right)$.

4 Setup Instructions

The converter modeling has two main stages. In the first stage, the positron creation events are simulated with Geant, and the resulting positrons are binned into a histogram. In the second stage, fits are performed for $\frac{dx}{ds}$ and $\frac{dy}{ds}$. As such, we have developed two executables: one responsible for the first stage, `converter_simulation`, and one responsible for the second, `converter_fitter`.

4.1 Dependencies

To build and run the simulation proper, `converter_simulation`, you will need 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` installed, although this is already a requirement for a standard *Bmad* distribution.

To build and run the fitting program, `converter_fitter`, you will need the GNU Scientific Library (GSL) installed on your system. This is distributed with *Bmad*, so you should already have it on your system.

Both executables require a C++ compiler with support for C++17. GCC 8.3 or higher should be fine; on the lab machines, you can get access to this GCC by adding

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

to your `.bashrc`.

4.1.1 Geant4 Installation Guide

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

1. Download the source files from https://geant4.web.cern.ch/support/download.
2. Create a directory where Geant will be installed. I'll be using `$HOME/geant`.
3. `cd` to this directory and unpack the downloaded files with

   ```
   $ tar xzvf ~/Downloads/geant4.10.06.tar.gz
   ```

   (change the version number as appropriate).
4. Make another directory where you will build Geant with

   ```
   $ mkdir geant4.10.06-build
   ```

5. `cd` to this new directory, and use `cmake` to configure the Geant4 build with

   ```
   cmake -DGEANT4_INSTALL_DATA=ON \
         -DCMAKE_INSTALL_PREFIX=$HOME/geant/geant4.10.06-install \
         ../geant4.10.06
   ```

   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. If you chose a different location for installing Geant, edit this flag as needed.

   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.

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

   ```
   $ make -jN
   ```

   where `N` is the number of threads you want to use for the compilation.

7. Once the compilation has finished, install to the install directory you specified in step 5 with

   ```
   $ make install
   ```

8. The file `$HOME/geant/geant4.10.06-build/geant4make.sh` must be sourced to add Geant4 to your path. To do so, add the following to your `.bashrc`:

   ```
   cd $HOME/geant/geant4.10.06-build && source geant4make.sh && cd -
   ```

   Again, if you chose a different location for installing Geant, modify this as necessary.

A build of Geant is already available on the lab machines at `/nfs/acc/temp/jmm699/geant`, so if you are working on the lab machines, you can simply add the following to your `.bashrc`:

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

4.2 Compiling the Executables

The converter simulation and fitting programs are distributed as part of the `util_programs` project, which is included in a standard *Bmad* distribution. If you are working with a release instead of a distribution, you can pull down the latest version of the `util_programs` project with

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

Once you have `util_programs` on your system, you will need to add

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

to your `.bashrc`, and then start a new shell. `cd` to the `util_programs` folder, and then simply run `mk` to build `converter_simulation` and `converter_fitter` (or `mkd` for debug builds).

5 How to run the programs

5.1 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 electron energies to be simulated
out_pc_min = 0 ! Minimum pc cutoff for outgoing positrons, defaults to 0
out_pc_max = 100000000 ! Maximum pc cutoff for outgoing positrons (in eV here)
dxy_ds_max = 10 ! Maximum cutoff for the magnitude of dx/ds
                ! and dy/ds allowed for outgoing positrons
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`.

5.2 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 positron 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.

5.3 The Fitting Program

Once the simulations are complete, just run

```
$ converter_fitter
```

in the same directory where you ran `converter_simulation`. `converter_fitter`will re-parse 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 9-15 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.