Code implementation of 2D random walks

Field Precision LLC
E mail: techinfo@fieldp.com
Internet: https://www.fieldp.com

We are developing **ProtoCalc**, an extension to the **Xenos** package, to model ion interactions with matter. **Xenos** currently handles only electrons and photons. A primary source of ion data is the **SRIM**[1] suite to find ion range and stopping power in materials for a given input energy. An additional output is the transverse straggling at the end of the trajectory, the RMS radius of the ion distribution about the projection of the initial orbit vector. An issue for **ProtoCalc** is how to represent the evolution of the transverse distribution over the length of the trajectory based on the end value. The process is a random walk in the transverse plane resulting from a large number of small angle scattering events. This report summarizes some basic results for a 2D random walk and how the process is implemented in **ProtoCalc** and **GenDist** (our utility for generating particle distributions). I used **GenDist** to confirm the theory numerically, to test algorithms and to check accuracy versus the number of steps.

In the limit of a large number of steps and particles, a one-dimensional random walk that starts from the origin converges to the normal probability distribution:

$$p(x)dx = \frac{dx}{\sigma_x\sqrt{2\pi}}\ \exp\left[-\frac{1}{2}\left(\frac{x}{\sigma_x}\right)^2\right]. \tag{1}$$

where σ_x is the root-mean-squared value of x. In a 2D walk from the origin in the transverse plane, the final distribution has cylindrical symmetry. If r is the distance from the origin, the probability distribution is:

$$p(r)dr = \frac{2rdr}{\sigma_r^2}\ \exp\left[-\left(\frac{r}{\sigma_r}\right)^2\right]. \tag{2}$$

Defining the variable $R = (r/\sigma_r)^2$, we can rewrite Eq, 2 as:

$$p(R)dR = \exp(-R)\ dR. \tag{3}$$

In contrast to the normal distribution of Eq. 1, Eq. 3 is integrable. It is easy to show that the probability function is normalized and to confirm that the expected value of R is

$$\overline{R} = \int_0^\infty\ R\ p(R)dR = 1.0. \tag{4}$$

Equation 4 implies that the expected root-mean-squared value of the radial displacement is $\overline{r^2} = \sigma_r^2$.

Suppose we initiate a random walk from the origin with random angle, uniform step size λ and N_s steps. When $N_s \gg 1$, the expectation is that

[1]Stopping and Range of Ions in Matter, http://www.srim.org

the distribution will approach that of Eq. 2. To generate an approximate distribution with specified σ_r, the step size should be:

$$\lambda = \frac{\sigma_r}{\sqrt{N_s}}. \tag{5}$$

I experimented with numerical routines and confirmed agreement with Eqs. 2, 4 and 5 using our program **GenDist** because it supports extensive diagnostics for distributions. A second motivation was that the work would add a new option for soft beam generation to to the program. In response to the script command

```
DEF CIRCGAUSS SigmaR Np [Ns],
```

GenDist initiates random walks for N_p particles that approach a the transverse probability distribution of Eq. 2 with the specified σ_r. I included the optional parameter N_s to check how many steps were necessary for a good approximation. Because the routines would be applied in **ProtoCalc**, I wanted a high speed algorithm that avoided redundant calculations of trigonometric functions. I defined a unit vector structure

```
TYPE UVector
  REAL x
  REAL y
END TYPE
```

and an array of the structure

```
TYPE (UVector) :: UVect(0:359)
```

At start up, **GenDist** fills the array with the values of the cosine and sin at angles $\theta_n = (0.017453)n$ where $n = 0, 1, ..., 359$. In the loop through random walk steps, direction vectors are chosen according to the calculated index

```
CALL Random_Number(Zeta)
N = INT(359.0*Zeta + 0.49)
```

where ζ is a random number in the range 0.0 to 1.0.

Figure 1 shows some results for parameters $N_p = 100,000$ and $R_{out} = \sigma_r = 1.0$. The top plots are histograms of the number of particles per radial interval and the bottom plots show the end distribution in the x-y plane of a limited number of the particles. The number of steps was $N_s = 10$ on the left and $N_s = 100$ on the right. There is little difference between the results. Numerical evaluations of the RMS radius gave 0.9991 for $N_s = 10$ and 0.9986 for $N_s = 100$. The implications are 1) that the method of choosing random

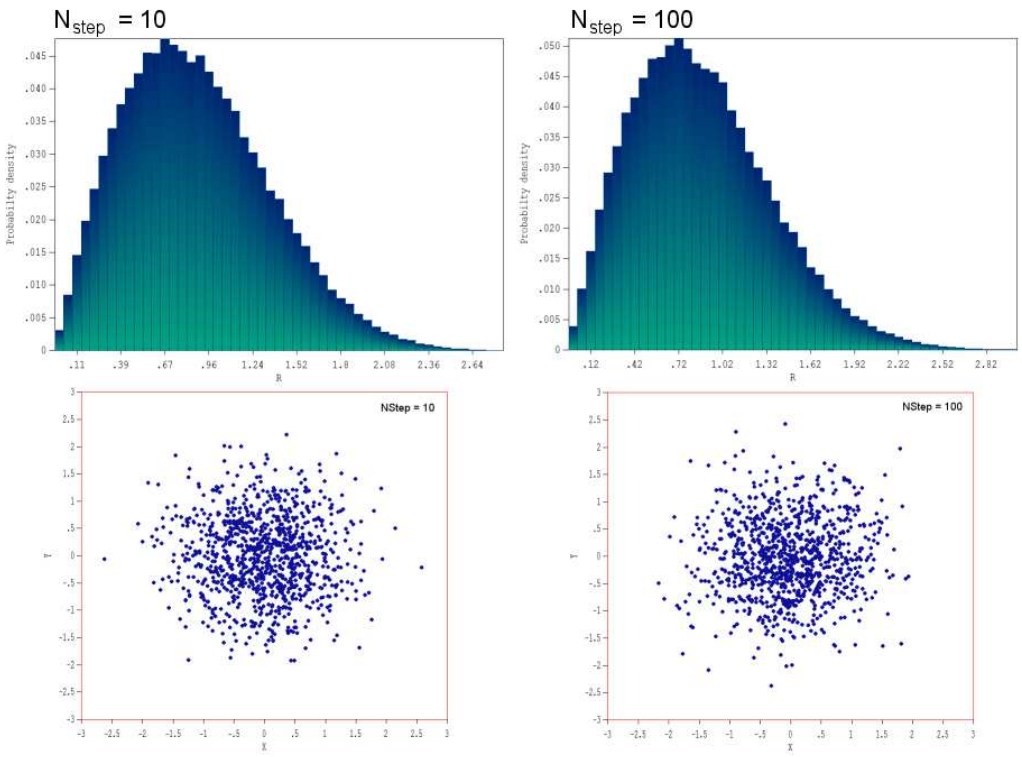

Figure 1: Comparison, two values of N_s for $N_p = 100,000$ and $R_{out} = \sigma_r = 1.0$. Top: radial probability distribution. Bottom: Distribution in the transverse plane, plot of 1/100 of the particles for clarity.

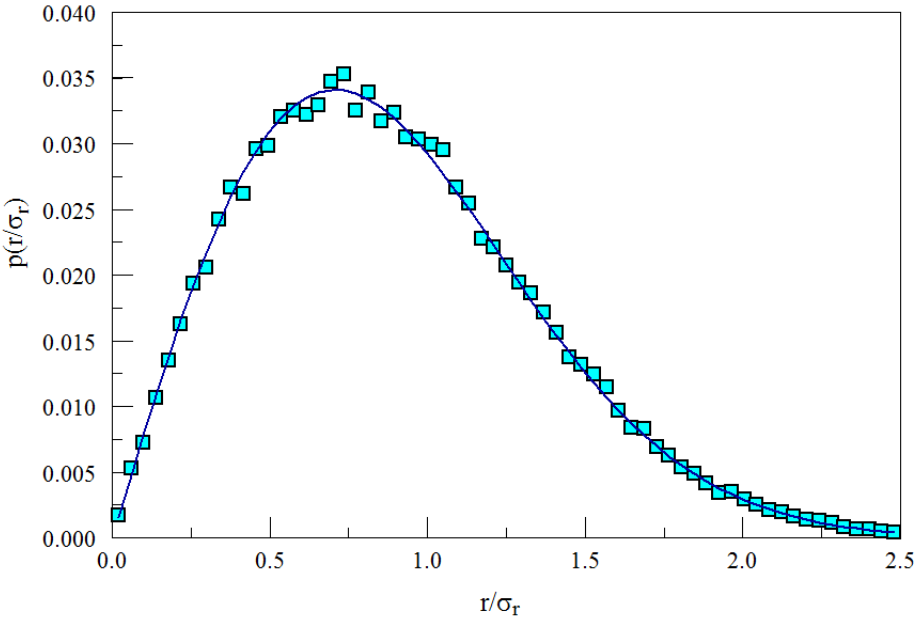

Figure 2: Numerical results (symbols) compared to the prediction of Eq. 2 (line).

step direction vectors does lead to a radially symmetric distribution and 2) that any value of N_s greater than 10 is sufficient for a good representation. Figure 2 further confirms the validity of the numerical method. The blue symbols are derived from the histogram of Fig. 1 with $N_s = 100$ and $\sigma_r = 1.0$ while the line is a plot of Eq. 2.

The application to **ProtoCalc** is straightforward for calculations with no applied magnetic field in a single material. The `PRT` file that defines the input beam distribution gives an initial position $\mathbf{X_0}$ and unit directional vector $\mathbf{U_0}$ for each ion. A calculation for an ion consists of tracking the trajectory in space by dividing the trajectory into a number N_s segments. The change in ion energy over the segment derived from the tabulated stopping power is assigned to the element occupied by the midpoint of the segment. The number of segments is determined from the range R_n and the uniform segment length D:

$$N_s = \frac{R_n}{D}. \tag{6}$$

For a smooth distribution of energy on the geometric mesh, D should be less than or equal to about half the minimum element dimension. While moving over segments, the ion performs a transverse random walk with step length λ given by Eq. 5. The axial step size is adjusted to

$$D \Rightarrow \sqrt{D^2 - \lambda^2}, \tag{7}$$

so that the total ion path equals R_n. The ion path projected on the $\mathbf{U}_0$ direction consists of a set of $(N_s + 1)$ points separated by D starting at $\mathbf{X_0}$.

The final issue is find the actual trajectory that includes the transverse displacements of the random walk. From the **GenDist** study, we know the answer in a frame of reference with origin at $\mathbf{X_0}$ where the z' axis points along $\mathbf{U_0}$. The x' and y' axes are normal to z' and orthogonal. The orientation of the x'-y' plane is arbitrary because there is no preferred transverse coordinate system for the random walk. After n steps, the axial position is $z'_n = nD$ while the transverse positions follow from the random walk algorithm, x'_n and y'_n. To transfer to the simulation frame, we find two orthogonal unit vectors normal to $\mathbf{U_0}$ by taking cross products, $\mathbf{U_{x0}}$ and $\mathbf{U_{y0}}$. The position at the end of segment n is

$$\mathbf{X}_n = \mathbf{X}_0 + nD\mathbf{U}_0 + x'_n\mathbf{U}_{x0} + y'_n\mathbf{U}_{y0}. \tag{8}$$

The trajectory calculation in **ProtoCalc** will be extremely fast. The vectors in Eq. 8 are evaluated once at the start of a track while the random walk displacements are determined by multiplication and addition operations.