Lecture notes:
Efficiency, statistics, and sampling

Alex F Bielajew
Institute for National Measurement Standards
National Research Council of Canada
Ottawa, Canada
K1A 0R6

Tel: 613-993-2715
FAX: 613-952-9865
e-mail: alex@irs.phy.nrc.ca

National Research Council of Canada Report PIRS-0395

Sir,
In your otherwise beautiful poem (The Vision of Sin) there is a verse which reads

> *"Every moment dies a man,*
> *every moment one is born."*

Obviously, this cannot be true and I suggest that in the next edition you have it read

> *"Every moment dies a man,*
> *every moment* $1\frac{1}{16}$ *is born."*

Even this value is slightly in error but should be suffuciently accurate for poetry.

...Charles Babbage (in a letter to Lord Tennyson)

1 Introduction

In this lecture we discuss some of the elements of the Monte Carlo technique. We consider some of the basic constituents that are essential to any Monte Carlo calculation, specifically, the generation of random numbers, the use of statistics that determine the quality of the calculated results, the definition of efficiency of a calculation and elementary sampling theory. The lecture concludes with a selection of examples of selecting from probability distributions.

2 Pseudo Random Number Generators

2.1 State-of-the-art *ca* 1987

The "pseudo" random number generator (RNG) is the "soul" of a Monte Carlo calculation. It is what generates the pseudo-random nature of Monte Carlo simulations thereby imitating the true stochastic nature of particle interactions. Consequently, much mathematical study has been devoted to RNG's [1, 2, 3]. These three references are excellent reviews of RNG theory and methods up to about 1987.

The operative phrase to be used when considering RNG's is "use extreme caution". <u>DO USE</u> an RNG that is known to work well and is widely tested. <u>DO NOT FIDDLE</u> with RNG's unless you understand thoroughly the underlying mathematics and have the ability to test the new RNG thoroughly. <u>DO NOT TRUST</u> RNG's that come bundled with standard mathematical packages. For example, DEC's `RAN` RNG (a system utility) and IBM's `RANDU` (part of the SSP mathematical package) are known to give strong triplet correlations. This would affect, for example, the "random" seeding of an isotropic distribution of point sources in a 3-dimensional object. A picture of an artefact generated by these RNG's s given in the Lecture "Running EGS4 on different architectures".

The gathering of random numbers into planes is a well-known artefact of RNG's. Marsaglia's classic paper [4] entitled "Random Numbers fall mainly in the Planes", describes how random numbers gather into $(n-1)$-dimensional hyperplanes in n-space for $n > 2$. Good RNG's either maximise the number of planes that are constructed to give the illusion of randomness or eliminate this artefact entirely. One must be aware of this behaviour in case anomalies do occur.

We provide two standard RNG's for use with EGS, an IBM and a VAX version. To our knowledge, no one has yet reported any anomalous results from these RNG's. The IBM version takes the form:

```
:Initialisation:   INTEGER IX(2);
                   REAL*8 DRN;
                   EQUIVALENCE (IX(1),DRN);
                   DATA IX(1)/Z46000000/;
                   IXX = 987654321;

:Iteration:        IXX = IXX * 663608941;
                   IX(2) = IXX;
                   R = DRN + 0.0D0;
```

This coding takes advantage of the IBM-specific hardware implementation of the manipulation of unnormalised floating point numbers. The changes in `IX(2)` change the lower order bits of `DRN` while the fixed group of higher order bits established by `IX(1)` guarantees the normalisation over the range $[0,1]$. (The addition with zero in the last line normalises the real floating point number.)

This is a very fast RNG taking only 4 fetches, 3 stores, one integer multiplication, one floating point multiplication, one floating point addition, and one type conversion from double to single precision (which is not always required). The disadvantage of this routine is that it is not transportable to other computer systems.

An equivalent but more transportable and slower routine is the VAX version:

```
:Initialisation:    IXX = 987654321;

:Iteration:         IXX = IXX * 663608941;
                    R = 0.5 + IXX * 0.23283064E-09;
```

This routine uses 4 fetches, 2 stores, 2 floating point operations, and one integer to real type conversion. This routine is transportable to all machines with a 32-bit, 2's-complement integer representation.

These RNG's are the most heavily used computer coding in EGS. In order to save the overhead of unnecessary calls to subroutines (20 to 1000 machine cycles), they are not used as subroutines but are inserted "in-line" wherever they are needed.

Good RNG's have large cycles. The cycle of a RNG is the number or random numbers it produces before repeating. The cycle of our RNG is 2^{30}, which is the theoretical upper limit for this class of RNG. It is essential, however, that the RNG be seeded with an odd number, otherwise the cycle would be 1! This is because a multiplication by 2 is equivalent to a left bit shift. At most 32 iterations would be needed to cause IXX to become 0. At this point it remains 0 for all remaining interactions. Note that the modification of IXX causes an overflow as the unnecessary bits "spill" over the left side. Consequently, integer overflow/underflow trapping should be disabled when using these RNG's.

2.2 New Random Number Generators

Recently, new forms of random number generators based on lagged-Fibonacci sequences have been devised [5, 6] that have extremely long repeat sequences (up to 2^{926} real numbers with 24-bit fractions!). A "universal" RNG (with a period length of 2^{144}) based on this approach has been proposed that has been shown to produce identical sequences on the entire spectrum of computers, from IBM PC's (kFLOP's) to ETA supercomputers (GFLOP's).

This is the future direction of RNG's and is discussed further in the Lecture "Running EGS4 on different architectures".

3 Statistical analysis

3.1 Estimating errors

Assume that x is a quantity we calculate during the course of a Monte Carlo simulation, *i.e.* a scoring variable. The output of a Monte Carlo calculation is usually useless unless we can ascribe a probable error to it.

The conventional approach to calculating the probable error is as follows:

- Assume that the calculation calls for N "incident" particle histories.
- Split the N histories into n statistical batches of N/n histories each. (*e.g.* We have chosen $n = 10$ as a standard for our calculations.) The calculated quantity for each of these batches is called x_i.
- Calculate the mean value of x:
$$\overline{x} = \frac{1}{N}\sum_{i=1}^{n} x_i \tag{1}$$
- Estimate the variance associated with the distribution of the x_i:
$$s_x^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i^2 - \overline{x}^2) \tag{2}$$
- The estimated variance of $\overline{x}$ is the standard variance of the mean:
$$s_{\overline{x}}^2 = \frac{s_x^2}{n} \tag{3}$$
(It is the error in $\overline{x}$ we are seeking, not the "spread" of the distribution of the x_i.)
- Report the final result as $x = \overline{x} \pm s_{\overline{x}}$.

Remarks:

The derivation of eq. 3 assumes that the x_i are normally distributed about $\overline{x}$. This is an assumption. Yet, we use eq. 3 with $n = 10$ because it still gives a reasonable estimate of the error in $\overline{x}$. In reality, decisions based upon the error in $\overline{x}$ are usually subjective in nature. We have found that when the errors are large ($> 5\%$ or so), the errors tend to be underestimated. For smaller errors the values tend to be sensible. There is some evidence that the calculated statistic depends weakly on the choice of n (8 to 12% or so). Therefore, it is important to report how your statistics were done when you publish your Monte Carlo results.

3.2 Combining errors of independent runs

For m independent Monte Carlo runs, it is easy to derive the following relation:

$$\overline{x} = \sum_{j=1}^{m} \left(\frac{N_j}{N}\right) \overline{x}_j \tag{4}$$

where $\overline{x}_j$ is the value of $\overline{x}$ for the j^{th} run and N_j is the number of histories in the j^{th} run. The total number of histories is given by:

$$N = \sum_{j=1}^{m} N_j \tag{5}$$

Then, assuming 1$^{\mathrm{st}}$-order propagation of independent errors, it is also easy to derive:

$$s_{\overline{x}}^2 = \sum_{j=1}^{m} \left(\frac{N_j}{N}\right)^2 s_{\overline{x}_j}^2 \tag{6}$$

where $s_{\overline{x}_j}^2$ is the estimated variance in $\overline{x}_j$.

Example: For m = 2:

$$\overline{x} = \left(\frac{N_1}{N}\right) \overline{x}_1 + \left(\frac{N_2}{N}\right) \overline{x}_2 \tag{7}$$

$$N = N_1 + N_2 \tag{8}$$

$$s_{\overline{x}} = \sqrt{\left(\frac{N_1}{N}\right)^2 s_{\overline{x}_1}^2 + \left(\frac{N_2}{N}\right)^2 s_{\overline{x}_2}^2} \tag{9}$$

Remarks:

This method of combining errors effectively increases the value of n, the number of statistical batches used in the calculation. In view of the fact that the calculated statistics are thought to depend weakly on n, it is preferable (but only marginally so for the sake of consistency) to combine the x_i's (the raw data) into the standard number of statistical batches. This is easy to do by initialising the data arrays to the results of the previous run before the start of a new run.

4 Efficiency

See text at beginning of Lecture "Variance reduction techniques".

5 Elementary sampling theory

Having considered RNG's in some detail, we now relate random numbers (RN's) to physical variables that one must obtain from given probability distribution functions (PDF's).

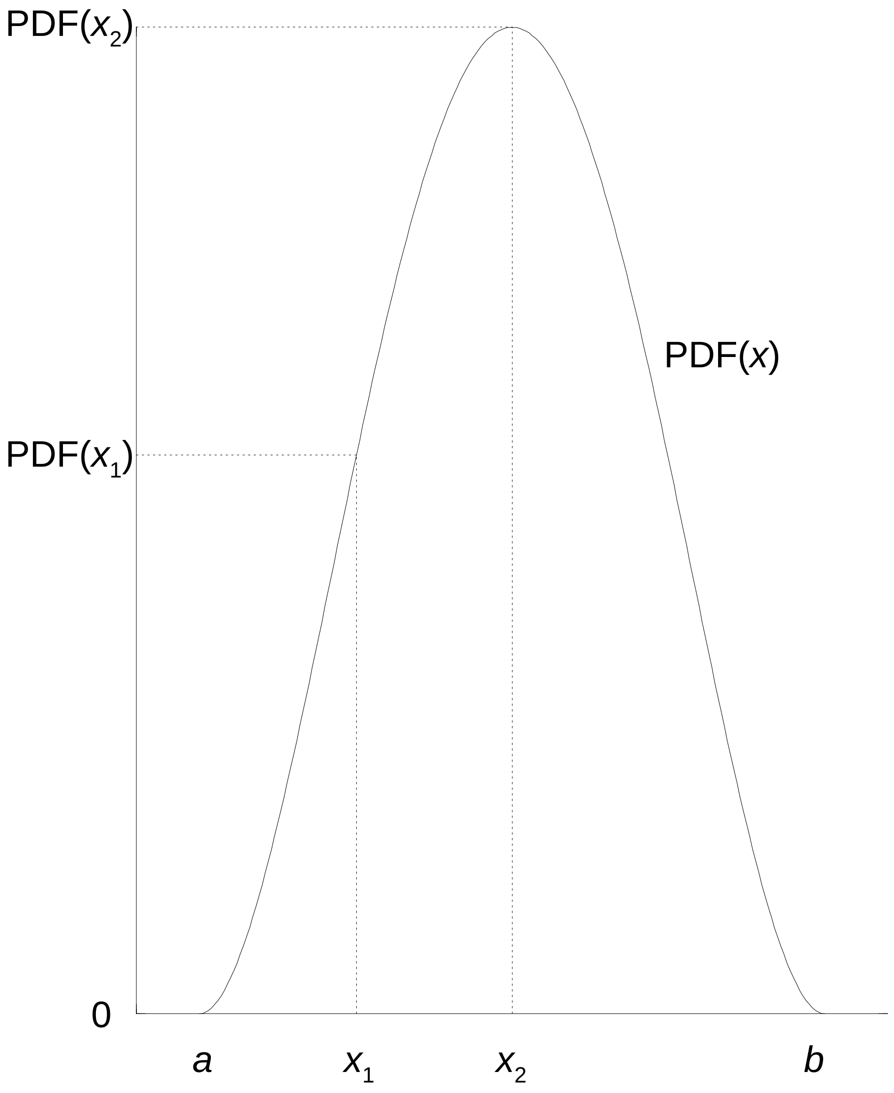

Figure 1: A typical probability distribution.

5.1 Invertible cumulative distribution functions (direct method)

A typical PDF is shown in fig. 1. It is defined over the range $[a, b]$ where neither a nor b are necessarily finite. A PDF *must* have the properties that it is integrable (so that one can normalise it by integrating it over its entire range) and that it is non-negative. (Negative probability distributions are difficult to interpret.)

We now construct its cumulative probability function (CDF):

$$r = \mathrm{CDF}(x) = \int_a^x \mathrm{d}x' \ \mathrm{PDF}(x') \tag{10}$$

and assume that it is properly normalised, *i.e.* $\mathrm{CDF}(b) = 1$. The corresponding CDF for our example is shown in fig. 2.

By its definition, we can map the CDF onto the range of random variables, r, where $0 \le r \le 1$. Now consider two equally spaced intervals $\mathrm{d}x_1$ and $\mathrm{d}x_2$, differential elements in x in the vicinity of x_1 and x_2. Using some elementary calculus we see that:

$$\frac{\mathrm{d}r_1}{\mathrm{d}r_2} = \frac{(\mathrm{d}/\mathrm{d}x)\mathrm{CDF}(x)|_{x=x_1}}{(\mathrm{d}/\mathrm{d}x)\mathrm{CDF}(x)|_{x=x_2}} = \frac{\mathrm{PDF}(x_1)}{\mathrm{PDF}(x_2)} \tag{11}$$

We can interpret this as meaning that, if we select many random variables in the range [0,1], then the number that fall within $\mathrm{d}r_1$ divided by the number that fall within $\mathrm{d}r_1$ is equal to the ratio of the probability distribution at x_1 to x_2.

Having mapped the random numbers onto the CDF, we may invert the equation to give:

$$x = \mathrm{CDF}^{-1}(r) \tag{12}$$

All CDF's that arise from properly defined PDF's are invertible, numerically if not analytically.

Then, by choosing r's randomly over a uniform distribution and substituting them in the above equation, we generate x's according to the proper PDF.

Example:

The number of mean free paths (MFP's), z, to an interaction is governed by the well-known PDF:

$$\mathrm{PDF}(z) = e^{-z} \tag{13}$$

The valid range of z is $0 \le z < \infty$ and this PDF is already properly normalised. The corresponding CDF and its random number map is given by:

$$r = \mathrm{CDF}(z) = 1 - e^{-z} \tag{14}$$

Inverting gives:

$$z = -\log(1 - r) \tag{15}$$

If r is uniformly distributed over $[0, 1]$ then so is $1 - r$. An equivalent form of the above equation (that saves one floating point operation) is:

$$z = -\log(r) \tag{16}$$

This is exactly the form used to calculate particle MFP's in EGS.

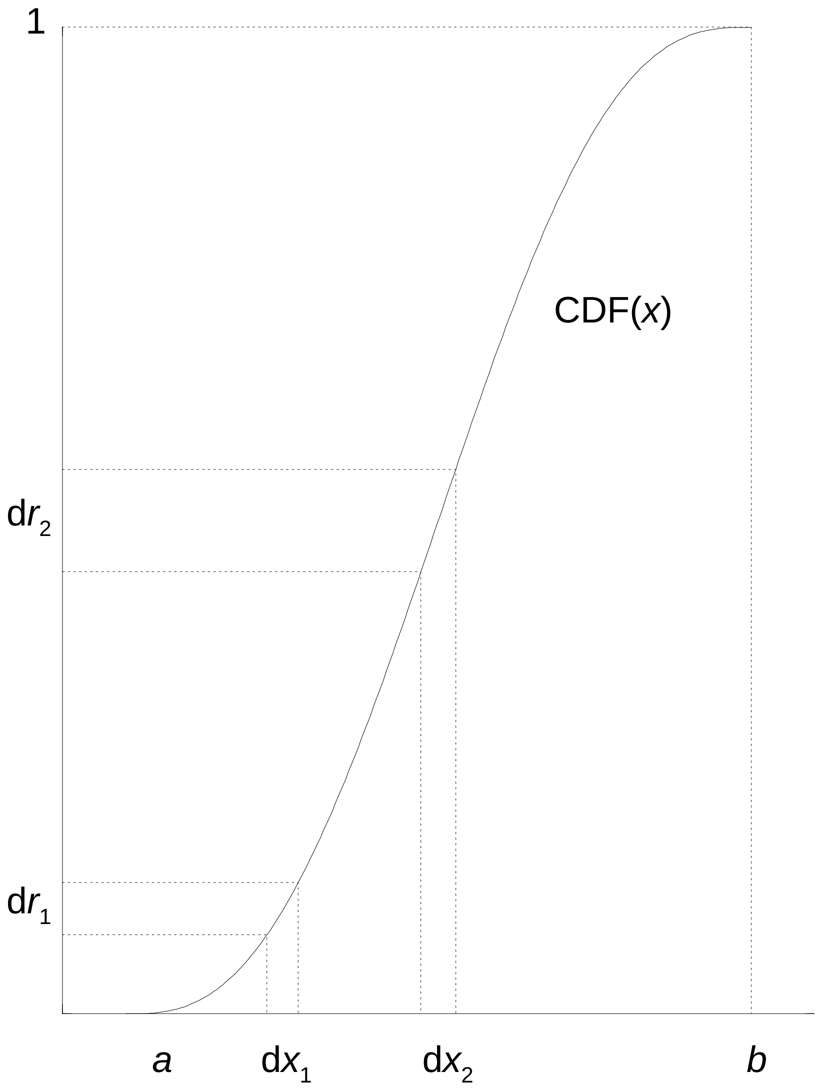

Figure 2: The cumulative probability distribution obtained by integrating the PDF in fig. 1.

5.2 Rejection method

While the invertible CDF method is always possible, at least in principle, it is often impractical to calculate CDF^{-1} because it may be exceedingly complicated mathematically. Another approach is to use the rejection method.

In recipe form, the procedure is this:

1. Scale the PDF by its maximum value obtaining a new PDF, $\mathrm{PDF}' = \mathrm{PDF}/\mathrm{PDF}_{\mathrm{max}}$, which has a maximum value of 1 (see figs. 3 and 4). Clearly, this method works only if the PDF is not infinite anywhere and if it is not prohibitively difficult to determine the location of the maximum value.

2. Choose a random number, r_1, uniform in the range $[0, 1]$ and use it to obtain an x which is uniform in the PDF's range [a,b]. (To do this, calculate $x = a + (b - a)r_1$.) (Note: This method is restricted to finite values of a and b. However, if either a or b are infinite a suitable transformation may be found to allow one to work with a finite range. *e.g.* $x \in [a, \infty)$ may be mapped into $y \in [0, 1)$ via transformation $x = a[1 - \log(1 - y)]$.)

3. Choose a second random number r_2. If $r_2 < \mathrm{PDF}(x)/\mathrm{PDF}_{\mathrm{max}}$ (region under $\mathrm{PDF}(x)/\mathrm{PDF}_{\mathrm{max}}$ in fig. 4) then accept x, else, reject it (shaded region above $\mathrm{PDF}(x)/\mathrm{PDF}_{\mathrm{max}}$ in fig. 4) and go back to step 2.

Remarks:

This method will result in x being selected according to the PDF. Some consider this method "crude" because random numbers are "wasted" unlike the invertible CDF method. It is particularly wasteful for "spiked" PDF's. However, it can save computing time if the CDF^{-1} is very complicated. One has to "waste" many random numbers to use as much computing time as in the evaluation of a transcendental function!

5.3 Mixed methods

As a final topic in elementary sampling theory we consider the "mixed method", a combination of the previous two methods.

Imagine that the PDF is too difficult to integrate and invert, ruling out the direct approach without a great deal of numerical analysis, and that it is "spiky", rendering the rejection method inefficient. (Many probability distributions have this objectionable character.) However, imagine that the PDF can be factored as follows:

$$\mathrm{PDF}(x) = f(x)g(x) \tag{17}$$

where $f(x)$ is an invertible function that contains most of the "spikiness", and $g(x)$ is relatively flat but contains most of the mathematical complexity.
The recipe is as follows:

1. Normalise $f(x)$ producing $\tilde{f}(x)$ such that $\int_a^b \mathrm{d}x\ \tilde{f}(x) = 1$.

2. Normalise $g(x)$ producing $\tilde{g}(x)$ such that $\tilde{g}(x) \leq 1\ \forall\ x \in [a, b]$.

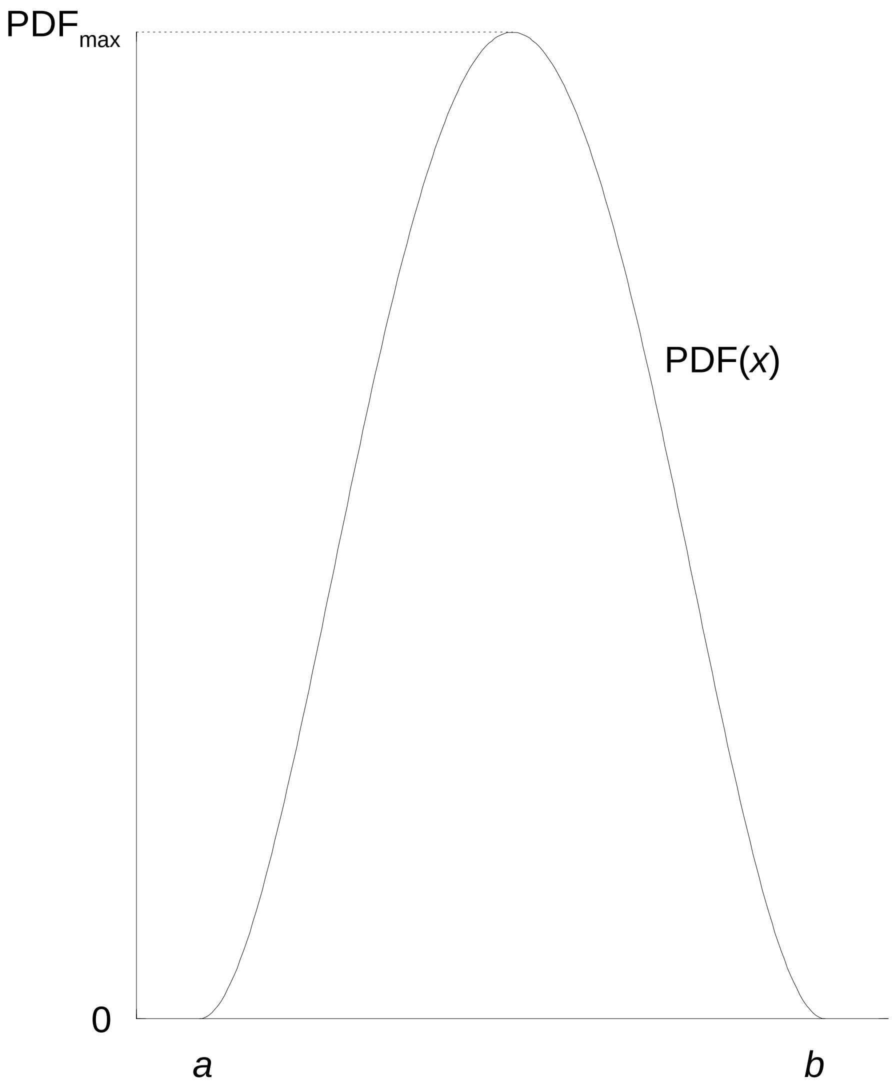

Figure 3: A typical probability distribution.

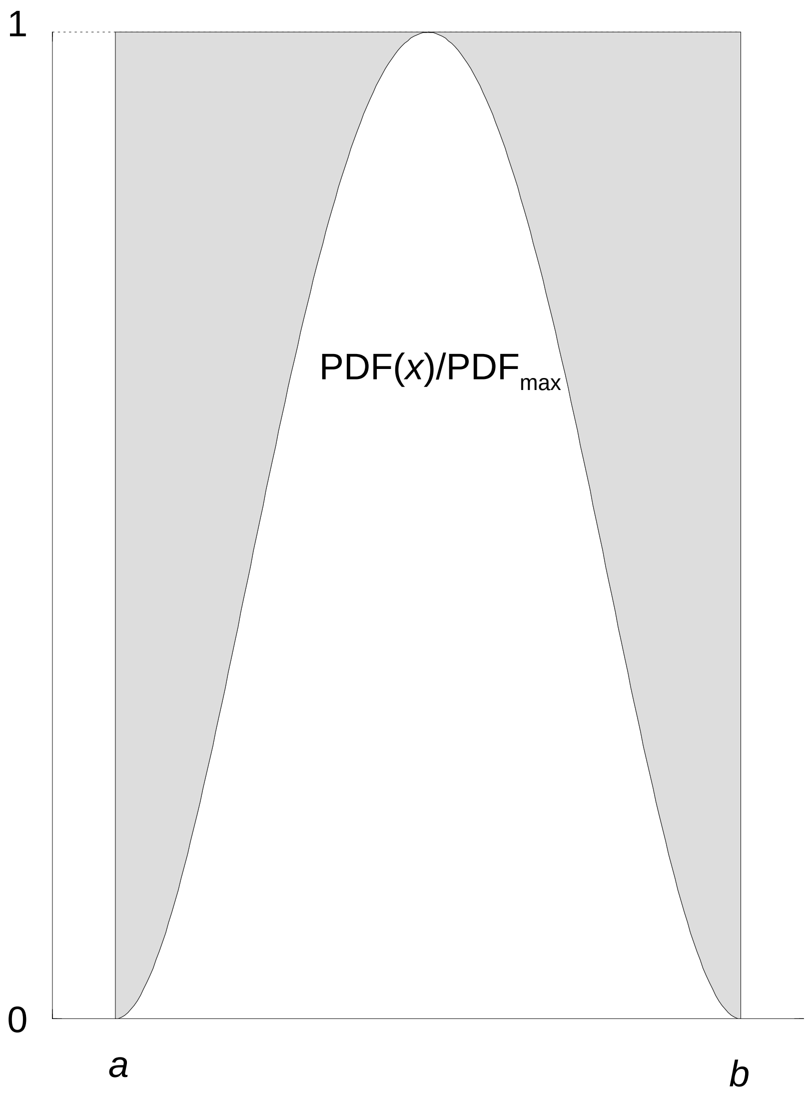

Figure 4: The probability distribution of fig. 3 scaled for the rejection technique.

3. Using the direct method described previously, choose an x using $\tilde{f}(x)$ as the PDF.

4. *Using this* x, apply the rejection technique using $\tilde{g}(x)$. That is, choose a random number, r, uniformly in the range $[0,1]$. If $\tilde{g}(x) \le r$, accept x, otherwise go back to step 3.

Remarks:

With some effort, any mathematically complex, spiky function can be factored in this manner. The art boils down to the appropriate choice of $\tilde{f}(x)$ that leaves a $\tilde{g}(x)$ that is nearly flat. For two recent examples of this method as applied to the EGS4 code, see refs. [7] and [8].

5.4 Examples of sampling techniques

5.4.1 Circularly collimated parallel beam

The normalised probability distribution in this case is:

$$\mathrm{d}^2 p(\rho,\phi) = \frac{1}{\pi\rho_0^2}\rho\mathrm{d}\rho\mathrm{d}\phi \qquad 0 \le \rho \le \rho_0^2 \quad 0 \le \phi \le 2\pi \tag{18}$$

where ρ is the cylindrical radius, ρ_0 is the collimation radius and ϕ is the azimuthal angle. $\rho\mathrm{d}\rho\mathrm{d}\phi$ is a differential surface element in cylindrical coordinates. This is a separable probability distribution of the form:

$$\mathrm{d}^2 p(\rho,\phi) = \mathrm{d}p_1(\rho)\mathrm{d}p_2(\phi) \tag{19}$$

where:

$$\mathrm{d}p_1(\rho) = \frac{2}{\rho_0^2}\rho\mathrm{d}\rho \qquad 0 \le \rho \le \rho_0 \tag{20}$$

and

$$\mathrm{d}p_2(\phi) = \frac{1}{2\pi}\mathrm{d}\phi \qquad 0 \le \phi \le 2\pi \tag{21}$$

<u>**Direct method**</u>

The CDF's in this case are:

$$\mathrm{CDF}_1(\rho) = c_1(\rho) = \frac{2}{\rho_0^2}\int_0^\rho \mathrm{d}\rho'\rho' = \frac{\rho^2}{\rho_0^2} \tag{22}$$

$$\mathrm{CDF}_2(\phi) = c_2(\phi) = \frac{1}{2\pi}\int_0^\phi \mathrm{d}\phi' = \frac{\phi}{2\pi} \tag{23}$$

Inverting gives:

$$\rho = \rho_0\sqrt{\xi_1} \tag{24}$$

$$\phi = 2\pi\xi_2 \tag{25}$$

where the ξ_i are random numbers on the range $[0,1]$.

The code segment that would produce the input phase-space parameters required by EGS looks like:

```
.
.
.
$RANDOMSET xi_1; rho = rho_0  * sqrt(xi_1);
$RANDOMSET xi_2; phi = two_pi * xi_2;
x = rho * cos(phi);
y = rho * sin(phi);
.
.
.
```

Rejection method

In this technique, a point is chosen randomly within the square $-1 \le x \le 1; -1 \le y \le 1$. If this point lies within a circle with unit radius the point is accepted and the x and y values scaled by the collimation radius, ρ_0. The code segment that would produce the input phase-space parameters required by EGS looks like:

```
.
.
.
LOOP
[
    $RANDOMSET xi_1; x = 2 * xi_1 - 1;
    $RANDOMSET xi_2; y = 2 * xi_2 - 1;
    IF(x**2 + y**2 <= 1) EXIT;
]
x = rho_0 * x;
y = rho_0 * y;
.
.
.
```

Which is better?

Actually, both methods are equivalent mathematically. However, one or the other may have advantages in execution speed depending on other factors in the application. If the geometry is not cylindrically symmetric or all the scoring that is done does not make use of the inherent cylindrical symmetry, then the rejection method is about twice as fast as the direct method because the trigonometric functions are not employed in the rejection method. (The default EGS4 trigonometric approximations were not employed during this test and the timing data was obtained on a Sun Workstation.)

If the geometry is cylindrically symmetric and the scoring takes advantage of this symmetry, then the direct method is about 2–3 times faster because symmetry reduces the calculation to:

```
.
```

```
.
.
$RANDOMSET xi_1; x = rho_0  * sqrt(xi_1);
y = 0;
.
.
.
```

5.4.2 Point source collimated to a planar circle

The normalised probability distribution in this case is:

$$\mathrm{d}^2 p(\theta,\phi) = \frac{\mathrm{d}\phi}{2\pi}\frac{\sin\theta \mathrm{d}\theta}{1-\cos\theta_0} \qquad 0 \le \theta \le \theta_0 \quad 0 \le \phi \le 2\pi \tag{26}$$

where θ is the polar angle and ϕ is the azimuthal angle. $\sin\theta\mathrm{d}\theta\mathrm{d}\phi$ is a differential surface element in spherical coordinates. θ_0 is the collimation angle. In terms of the distance to the collimation place z_0 and the diameter of the collimation circle on this plane ρ_0, $\cos\theta_0 = z_0/\sqrt{z_0^2+\rho_0^2}$.

This is a separable probability distribution of the form:

$$\mathrm{d}^2 p(\theta,\phi) = \mathrm{d}p_1(\theta)\mathrm{d}p_2(\phi) \tag{27}$$

where:

$$\mathrm{d}p_1(\theta) = \frac{\sin\theta\mathrm{d}\theta}{1-\cos\theta_0} \qquad 0 \le \theta \le \theta_0 \tag{28}$$

and

$$\mathrm{d}p_2(\phi) = \frac{1}{2\pi}\mathrm{d}\phi \qquad 0 \le \phi \le 2\pi \tag{29}$$

The CDF's in this case are:

$$\mathrm{CDF}_1(\theta) = c_1(\theta) = \frac{1}{1-\cos\theta_0}\int_0^\theta \sin\theta'\mathrm{d}\theta' = \frac{1-\cos\theta}{1-\cos\theta_0} \tag{30}$$

$$\mathrm{CDF}_2(\phi) = c_2(\phi) = \frac{1}{2\pi}\int_0^\phi \mathrm{d}\phi' = \frac{\phi}{2\pi} \tag{31}$$

Inverting gives:

$$\cos\theta = 1 - \xi_1[1-\theta_0] \tag{32}$$

$$\phi = 2\pi\xi_2 \tag{33}$$

where the ξ_i are random numbers on the range $[0,1]$.

The code segment that would produce the input phase-space parameters required by EGS looks like:

```
.
.
.
$RANDOMSET xi_1                         ;
cos_theta = 1 - xi_1 * (1 - cos_theta_0);
theta     = acos(theta)                 ;
sin_theta = sin(theta)                  ;
```

```
$RANDOMSET xi_2    ;
phi = two_pi * xi_2;

u = sin_theta * cos(phi); "u is sin(theta)*cos(phi), x-axis direction cosine"
v = sin_theta * sin(phi); "v is sin(theta)*sin(phi), y-axis direction cosine"
w = cos_theta           ; "w is cos(theta)         , z-axis direction cosine"

x = z_0 * u/w;
y = z_0 * v/w;
.
.
.
```

In terms of the cylindrical coordinates on the collimation plane, eq. 32 becomes:

$$\frac{z_0}{\sqrt{\rho^2 + z_0^2}} = 1 - \xi_1 \left[1 - \frac{z_0}{\sqrt{\rho_0^2 + z_0^2}} \right] \tag{34}$$

which yields a value for ρ on the collimation plane.

In the small angle limit, $\theta_0 \longrightarrow 0$, the circularly collimated parallel beam result should be recovered. If one employs the small angle approximation, $\rho \ll z_0$ and $\rho \ll z_0$, eq. 34 obtains the result of eq. 24, *i.e.* $\rho = \rho_0\sqrt{\xi_1}$.

References

[1] J.R. Ehrman, *The Care and Feeding of Random Numbers*, SLAC VM Notebook, Module 18, SLAC Computing Services (1981).

[2] D.E. Knuth, *The art of computer programming, Vol. II*, (Addison Wesley, Reading Mass.) (1981).

[3] F. James, *A Review of Pseudorandom Number Generators*, CERN-Data Handling Division, Report DD/88/22 (1988).

[4] G. Marsaglia, *Random numbers fall mainly in the planes*, Nat. Acad. Sci. **61** 25 – 28 (1968).

[5] G. Marsaglia, A. Zaman and W.W. Tsang, *Toward a Universal Random Number Generator*, Statistics and Probability Letters **8** 35 – 39 (1990).

[6] G. Marsaglia and A. Zaman, *A New Class of Random Number Generators*, Annals of Applied Probability **1** 462 – 480 (1991).

[7] A.F. Bielajew and D.W.O. Rogers, *Photoelectron angular distribution in the EGS4 code system*, National Research Council of Canada Report PIRS-0058 (1986).

[8] A.F. Bielajew, R. Mohan and C.S. Chui, *Improved bremsstrahlung photon angular sampling in the EGS4 code system*, National Research Council of Canada Report PIRS-0203 (1989).