Chapter 2 Choosing Random Numbers from Distributions

2.1 Direct inversion

In the previous chapter, we learned how computers generate pseudo-random numbers uniformly in the domain (0,1). In practice, though we need make to choices based on non-uniform distributions over other domains and based on discrete distributions.

In the next few subsections, we will study several ways of making choices from other distributions and over other (sometimes not continuous) domains. We will discuss these in order of increasing difficulty:

- Choosing random numbers uniformly over a general domain (a,b)
- Making choices from discrete distributions.
- Choosing random numbers from continuous distributions directly.
- Choosing random numbers from continuous distributions by rejection.

In each of these cases we will use ξ as the symbol for the uniform deviate (i.e., random number supplied by the computer in the domain (0,1)). The methods we will study will simply transform one or more uniform deviates into a random variable distributed as desired.

Choosing a random number uniformly over a domain (a,b)

If the distribution is uniform and the only problem is that the domain is (a,b) instead of (0,1), a uniform deviate can be transformed to the new domain use the intuitive:

$$x = a + (b-a)\xi \tag{2-1}$$

That is, the uniform deviate is used as the fractional distance from the lower limit of the domain to the upper limit.

Example: Choose random number uniformly in domain $(0, 2\pi)$.
Answer: $x = 2\pi\xi$

Example: Choose random number uniformly in domain (-1,1):
Answer: $x = 2\xi - 1$

The corresponding PDF is:

$$\pi(x) = \frac{1}{b-a} \tag{2-2}$$

Making choices from a discrete distribution

For the case that we must choose among *N* items, each of which has a relative probability of $\tilde{\pi}_i$, the basic idea is that we divide the domain (0,1) into appropriately sized sub-domains, pick a ξ, and then determine which sub-domain it falls into. A procedure to follow this is:

1. Normalize the probabilities by finding π_i using:

$$\pi_i = \frac{\tilde{\pi}_i}{\sum_{j=1}^{I} \tilde{\pi}_j} \tag{2-3}$$

2. Choose the item j as the integer that satisfies the relation:

$$\sum_{i=1}^{j-1} \pi_i < \xi < \sum_{i=1}^{j} \pi_i \tag{2-4}$$

Example: Choose between three items with relative probability of 1, 2, and 5.
Answer: Step one gets us from:

$$\tilde{\pi}_1 = 1; \tilde{\pi}_2 = 2; \tilde{\pi}_3 = 5$$

to

$$\pi_1 = 0.125; \pi_2 = 0.25; \pi_3 = 0.625$$

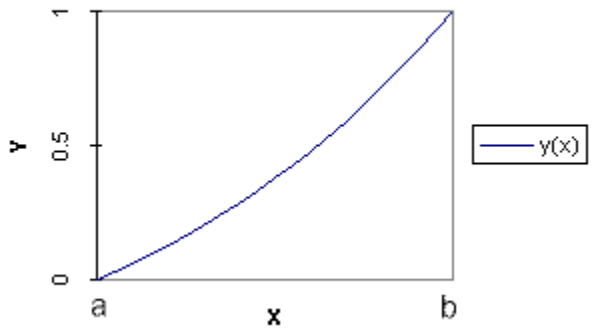
Therefore, based on a given ξ, we will choose:

Item 1 if $0 < \xi < 0.125$
Item 2 if $0.125 < \xi < 0.375$
Item 3 if $\xi > 0.375$

Choosing random numbers from a continuous distribution (directly)

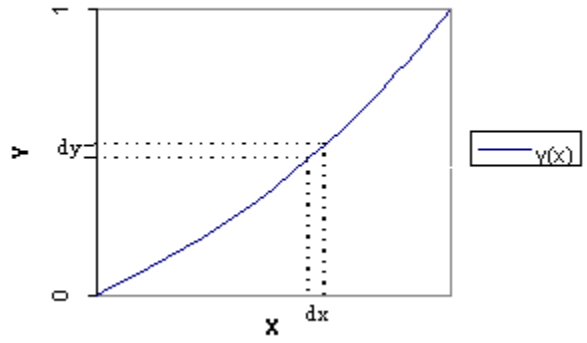
For non-uniform continuous distributions, we will show two methods. The first, in this subsection, is the preferred one for well-behaved functions that can be integrated and inverted easily.

Mathematically, we proceed by assigning the **x-axis** to the variable to be selected using a normalized probability distribution function $\pi(x)$ over a domain (a,b) and we let the **y-axis** be the pseudo-random number supplied by the computer. Then we set up a mapping function y=y(x) that relates the two. Ultimately, though, we will have to invert this to get a mapping from y to x. Okay, so it basically looks like this:



Note that the mapping function must be unique both in the x => y mapping and the y => x direction, so it must be a non-decreasing function.

To figure out the mapping function, we consider the unique mapping between two differential distances along the axes, dy and dx:



Since the two represent the same region of the curve, all *x* values inside *dx* will map into *dy* and vice versa, so they must correspond to the same probability:

$$\Pr\{y \text{ chosen in } dy\} = \Pr\{x \text{ chosen in } dx\}$$

$$dy = \pi(x)dx \qquad (2\text{-}5)$$

where $\pi(x)$ is the probability distribution in x and the corresponding (uniform) distribution in y is 1.

Solving for the function *y(x)* by integrating *x* from *a* to *x* and y from 0 to *y* gives:

$$y(x) = \int_a^x \pi(x')\,dx' \equiv \Pi(x), \tag{2-6}$$

where the last equality is added because the integral over *y* corresponds to our previous definition of the CDF. We see that the x => y mapping function must be the same as the Cumulative Distribution Function, $\Pi(x)$. So, to get this to by y => x, we must simply use our algebra skills to solve the above equation for *x*. (Good luck with that, by the way. You will likely find out the hard way that your Algebra teacher always fed you problems designed to work out nicely; real life is often messier.)

A step-by-step procedure for choosing an x from an unnormalized function $\tilde{\pi}(x)$ in the domain (a,b) is:

1. Form a normalized probability distribution function (PDF), *π(x)*, using:

$$\pi(x) = \frac{\tilde{\pi}(x)}{\int_a^b \tilde{\pi}(x')\,dx'}$$

2. Find the cumulative distribution function (CDF), $\Pi(x)$, using:

$$\Pi(x) = \int_a^x \pi(x')\,dx'$$

   [NOTE: Reality check. As a practical matter, I usually check my work at this point by making sure that Π(a)=0 and Π(b)=1.]

3. Set random number ξ equal to $\Pi(x)$:

$$\xi = \Pi(x)$$

4. Solve for x as a function of ξ:

$$x = \Pi^{-1}(\xi)$$

Therefore, given a uniform deviate, ξ_i, the sample $\hat{x}_i$ chosen from:

$$\hat{x}_i = \Pi^{-1}(\xi_i) \tag{2-7}$$

will be distributed according to $\pi(x)$.

Example: Choose a random number distributed according to $\tilde{\pi}(x) = e^x$ in the domain $(1, 2)$.

Answer: Step 1. Normalize the function:

$$\pi(x) = \frac{e^x}{\int_1^2 e^{x'} dx'} = \frac{e^x}{e^2 - e^1}$$

Step 2. Find the CDF:

$$\Pi(x) = \frac{e^x - e^1}{e^2 - e^1}$$

[NOTE: This passes the reality check since Π(1)=0 and Π(2)=1.]

Step 3. Set the CDF to a random number:

$$\xi = \frac{e^x - e^1}{e^2 - e^1}$$

Step 4. Solve for *x*:

$$\xi(e^2 - e^1) = e^x - e^1$$

$$e^x = \xi(e^2 - e^1) + e^1$$

$$x = \ln(e^1 + \xi(e^2 - e^1))$$

Therefore, *x* chosen using this formula will be distributed according to $\tilde{\pi}(x) = e^x$ in the domain $(1, 2)$.

Testing the result

There are many ways to check that the desired distribution is being reproduced by your resulting sequence of selected variables. The most satisfying is to choose *N* samples of x, then "bin" them into equal-sized divisions of the domain, and then check that a plot of the number of samples falling into each "bin" matches the (approximate) number that should fall into it. The Java code that I use for this task is reproduced below, where the method PDF(x) returns the PDF at the bin midpoint and Sample() implements the transformation of ξ into *x*. (The coding corresponds to the previous example problem.)

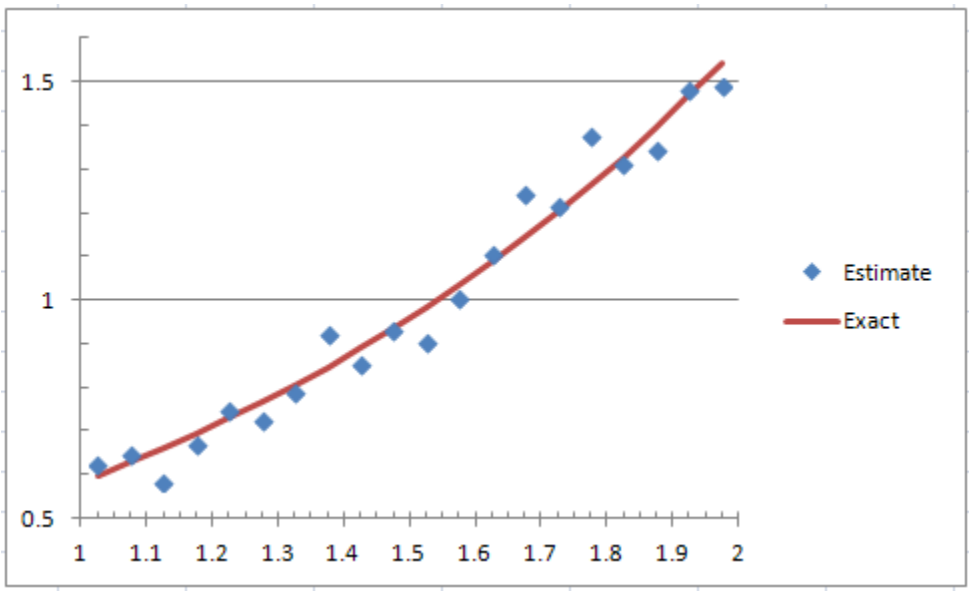
```
import java.util.Scanner;
class Bin
{
  public static void main(String[] args)
  {
    double a=1.;
    double b=2.;
    Scanner sc=new Scanner(System.in);
    while(true)
    {
      System.out.println(" Number of bins?");
      int nbin=sc.nextInt();
      if(nbin < 1)System.exit(0);
      double[] bin=new double[nbin];
      System.out.println(" Number of histories to run?");
      int N=sc.nextInt();
      double dx=(b-a)/nbin;
      for(int i=0;i<N;i++)
      {
        double x=Sample();
        int binNumber=(int)((x-a)/dx);
        bin[binNumber]+=1.;
      }
      double x=a-dx/2.;
      for(int i=0;i<nbin;i++)
      {
        x+=dx;
        bin[i]/=N*dx;
        System.out.printf(" Bin %1$5d Sample for x = %2$7.5f is %3$7.5f vs  %4$7.5f
Ratio (%5$f) \n",
        i,x,bin[i], PDF(x),bin[i]/PDF(x));
      }
    }
  }

  static double Sample()
  {
    double squiggle=Math.random();
    return Math.log(Math.exp(1.)+squiggle*(Math.exp(2.)-Math.exp(1.)));
  }

  static double PDF(double x)
  {
    return Math.exp(x)/(Math.exp(2.)-Math.exp(1.));
  }
}
```

Example: Applying this test procedure to the previous example, with 20 bins and using EXCEL to plot the results gives the following tables and plot:

```
Number of bins? 20
Number of histories to run? 10000
Bin     0 Sample for x = 1.02500 is 0.62600 vs  0.59671 Ratio (1.049087)
Bin     1 Sample for x = 1.07500 is 0.64800 vs  0.62730 Ratio (1.032993)
Bin     2 Sample for x = 1.12500 is 0.58600 vs  0.65947 Ratio (0.888598)
Bin     3 Sample for x = 1.17500 is 0.67200 vs  0.69328 Ratio (0.969309)
Bin     4 Sample for x = 1.22500 is 0.75000 vs  0.72882 Ratio (1.029057)
Bin     5 Sample for x = 1.27500 is 0.72600 vs  0.76619 Ratio (0.947545)
Bin     6 Sample for x = 1.32500 is 0.79200 vs  0.80547 Ratio (0.983272)
Bin     7 Sample for x = 1.37500 is 0.92400 vs  0.84677 Ratio (1.091204)
Bin     8 Sample for x = 1.42500 is 0.85400 vs  0.89019 Ratio (0.959350)
Bin     9 Sample for x = 1.47500 is 0.93200 vs  0.93583 Ratio (0.995911)
Bin    10 Sample for x = 1.52500 is 0.90600 vs  0.98381 Ratio (0.920912)
Bin    11 Sample for x = 1.57500 is 1.00600 vs  1.03425 Ratio (0.972687)
Bin    12 Sample for x = 1.62500 is 1.10600 vs  1.08728 Ratio (1.017221)
Bin    13 Sample for x = 1.67500 is 1.24400 vs  1.14302 Ratio (1.088344)
Bin    14 Sample for x = 1.72500 is 1.21800 vs  1.20163 Ratio (1.013627)
Bin    15 Sample for x = 1.77500 is 1.37800 vs  1.26323 Ratio (1.090851)
Bin    16 Sample for x = 1.82500 is 1.31200 vs  1.32800 Ratio (0.987951)
Bin    17 Sample for x = 1.87500 is 1.34600 vs  1.39609 Ratio (0.964122)
Bin    18 Sample for x = 1.92500 is 1.48400 vs  1.46767 Ratio (1.011127)
Bin    19 Sample for x = 1.97500 is 1.49000 vs  1.54292 Ratio (0.965703)
```



(Of course, most of your results will be much closer than this, but this is just an example.)

2.2 Rejection method

A second way of picking a variable x from a continuous distribution is the **rejection method**. It is less efficient and less elegant than the direct method, but is needed in cases where the direct method cannot be used. This can occur, of course, for a computer program if:

- The distribution is not pre-determined (e.g., input by user or created as the program runs).
- The PDF cannot be integrated.
- The CDF cannot be solved for *x*.

However, you must be able to determine an upper bound of the PDF.

The method is similar to the approach we took in the first Monte Carlo exercise we did, finding π by picking points inside an enclosing square but "scoring" only the ones inside the circle. Basically, we do the same thing: create a uniform distribution that contains (i.e., is everywhere) the desired function $\tilde{\pi}(x)$, pick an (x,y) point randomly inside the rectangle (i.e., under the bounding function), then keep it only if it is also under $\tilde{\pi}(x)$. It is not even necessary to normalize the function first.

A step-by-step procedure for using this method to choose an x from an unnormalized function $\tilde{\pi}(x)$ in the domain (a,b) is:

1. Find a bounding value (the maximum value is best), $\tilde{\pi}_{\text{sup}}$, of the PDF in the domain desired.
2. Pick an x uniformly in the domain using

$$\hat{x} = a + (b-a)\xi_1$$

3. Choose a y uniformly between 0 and $\tilde{\pi}_{\text{sup}}$ using:

$$\hat{y} = \xi_2 \tilde{\pi}_{\text{sup}}$$

If $\hat{y} < \tilde{\pi}(\hat{x})$, keep the $\hat{x}$. Otherwise, repeat 2-3.

Notice a couple of important points (i.e., errors that students frequently make):

1. This process is completely followed—including all loops—EACH TIME you need a random number. To say this another way: Until you satisfy the IF test in Step 4, YOU HAVE NOT CHOSEN THE NEXT NUMBER. For example, if you are asked to run 1000 histories involving a rejection algorithm, then you must pick 1000 x values that satisfy step 4. If it takes you 1100 or 2000 or 2 million choices of x before you get 1000 that pass step 4, then you must run the 1100, 2000, or 2 million passes. (Yes, this creates an inefficiency that is built into the method—it is the price we pay for simplicity of programming and flexibility.)

2. The basic idea—which is a powerful one that is often used in Monte Carlo—is that the successful production of an *x* value involves an AND logic:

$$\pi(x)dx \equiv \Pr\{x \text{ falls in } (x, x+dx)\}$$
$$\sim \Pr\{x \text{ chosen in } (x, x+dx)\} \times \Pr\{x \text{ is kept} \,|\, x \text{ was chosen}\} \tag{2-8}$$

In the direct method, the x value chosen was always kept, so the second probability was always 100%. As a result the tough "shape" of $\pi(x)$ had to be included in the "x is chosen" step, so we had to resort to fairly complicated algorithms to make it come out right.

In a rejection method, we keep the first step simple (by choosing *x* from a much easier distribution) and incorporate the "troublesome" $\pi(x)$ shape into the second step, by using:

$$\Pr\{x \text{ chosen in } (x, x+dx)\} = \frac{1}{b-a}$$
$$\Pr\{x \text{ is kept}\} = \frac{\pi(x)}{\pi_{\max}} \tag{2-9}$$

3. The reason that we needed a proportionality (~) in the equation two back is because the two sub-probabilities do not multiply to get $\pi(x)$; they multiply to get $\frac{\pi(x)}{\pi_{\max}(b-a)}$. What this means in practice is that the term $\frac{1}{\pi_{\max}(b-a)}$ is the <u>efficiency</u> of the rejection method. This efficiency will always be less than one and represents the long-term probability that the chosen "test x" will pass the Step 4 IF test.

Notice that the first step—the choice of *x*—does not have to be <u>that</u> simple: We used the very simplest distribution, the uniform distribution. But any other easy-to-choose-from $\pi(x)$ will work (and is required if the domain of *x* is infinite).

Example: If you were tasked to choose from the distribution $\pi(x) = \sin(x)e^{-x}$ over a given domain (a,b), you would immediately recognize this as a hard function to integrate and invert. For a rejection algorithm, you might try either of these three:

Option 1: Choose x uniformly in (a,b) and keep it with probability proportional to $\sin(x)e^{-x}$.

Option 2: Choose x in (a,b) according to (normalized) $\sin(x)$ and keep it with probability proportional to e^{-x}.

Option 3: Choose x in (a,b) according to (normalized) e^{-x} and keep it with probability proportional to $\sin(x)$.

In practice, the developer would probably use the option that is most efficient.

2.3 Probability mixing

The probability mixing method is a mathematical technique used for choosing random numbers from linear combinations of PDF's. It gets its name from the fact that it allows the user to "mix" different PDF's additively. The method shows up in many different forms, so in this lesson we will present the simple mathematics and then proceed to show several of the forms that it can show up in.

Mathematical form

The basic idea of the probability mixing method is that if you have a PDF (possibly unnormalized) that is the sum of other functions:

$$\begin{aligned}\tilde{\pi}(x) &= \tilde{\pi}_1(x) + \tilde{\pi}_2(x) + \ldots + \tilde{\pi}_N(x) \\ &= \sum_{n=1}^{N} \tilde{\pi}_n(x)\end{aligned} \tag{2-10}$$

over a domain (a,b) and **all** of the $\tilde{\pi}_n(x)$ are greater than zero in the domain (a,b). Then you can choose a random number between (a,b) according to $\tilde{\pi}(x)$ with a two-step procedure of:

1. Choosing one of the *N* subdistributions
2. Choosing x from that individual subdistribution.

The first choice is reduced to a discrete choice using each function's integral over the domain as its relative probability. (Note that this integral does double duty, serving both as the relative probability of choosing this subdistribution AND as the normalization fact that divides the original unnormalized subdistribution.)

The step-by-step procedure for doing this is:

1. For each of the *N* subdistributions, find its integral over the domain:

$$\tilde{\pi}_n = \int_a^b \tilde{\pi}_n(x)\, dx$$

2. Normalize the $\tilde{\pi}_n$'s so that they sum to 1:

$$\pi_n = \frac{\tilde{\pi}_n}{\sum_{i=1}^{N} \tilde{\pi}_i}$$

3. Choose one of the subdistributions, j, from 1 to N, from a discrete distribution using these π_n probabilities.
4. Choose a value of *x* using the chosen (now normalized) subdistribution:

$$\pi_j(x) = \frac{\tilde{\pi}_j(x)}{\tilde{\pi}_j}$$

Example: Sum of functions over entire domain
The most straight-forward application is when each of the subdistributions is defined over the entire domain. As an example, let us use

$\tilde{\pi}(x) = x^2 + e^{-x}$ over the domain (1,2).

[NOTE: If it were a minus sign between them, you couldn't do this!]

This, of course, can be broken down into:

$$\tilde{\pi}(x) = \tilde{\pi}_1(x) + \tilde{\pi}_2(x)$$

where:

$$\tilde{\pi}_1(x) = x^2$$
$$\tilde{\pi}_2(x) = e^{-x}$$

Following the procedure, we get:
Step1. Find the $\tilde{\pi}_n$'s:

$$\tilde{\pi}_1 = \int_1^2 x^2 dx = \frac{2^3 - 1^3}{3} = 2.3\overline{3}$$
$$\tilde{\pi}_2 = \int_1^2 e^{-x} dx = -(e^{-2} - e^{-1}) = 0.2325...$$

Step 2. Normalize the $\tilde{\pi}_n$'s:

$$\pi_1 = \frac{2.333\overline{3}}{2.333\overline{3} + 0.2325...} = 0.9094$$
$$\pi_2 = \frac{0.2325...}{2.333\overline{3} + 0.2325...} = 0.0906$$

Step 3. Choose one of the functions, j, using the π_n values in a discrete distribution:

We would do this by:

- Choosing a random number, ξ.
- If $\xi < 0.9094$, j=1.
- Otherwise, j=2.

Step 4: Choose a number x using the normalized distribution for the j chosen

- If j=1, this comes down to $x = \sqrt[3]{7\xi + 1}$ (Check this!)
- If j=2, then it is $x = -\ln(e^{-1} - 0.2325\xi)$ (Check this!)

The overall answer is that we choose from $\tilde{\pi}\ x\ = x^2 + e^{-x}$ over the domain (1,2) by choosing x from

$x = \sqrt[3]{7\xi + 1}$, 90.94% of the time

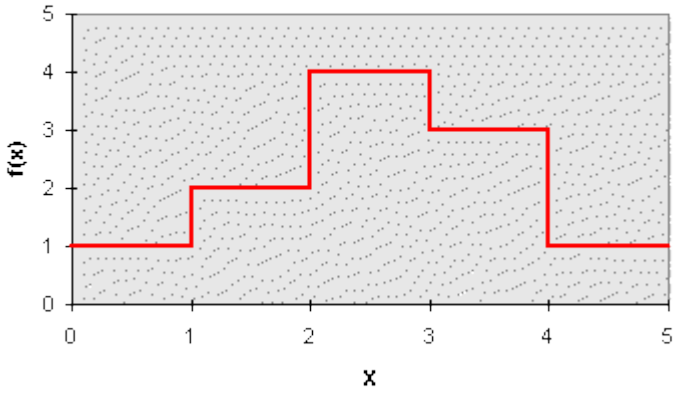
$x = -\ln(e^{-1} - 0.2325\xi)$, 9.06% of the time

Example: Histogram function

Another example is choosing a random number from a histogram function, which is a step function over contiguous regions:

$$\tilde{\pi}(x) = \begin{cases} f_1 \text{ for } a < x < x_1 \\ f_2 \text{ for } x_1 < x < x_2 \\ \vdots \\ f_N \text{ for } x_{N-1} < x < b \end{cases}$$

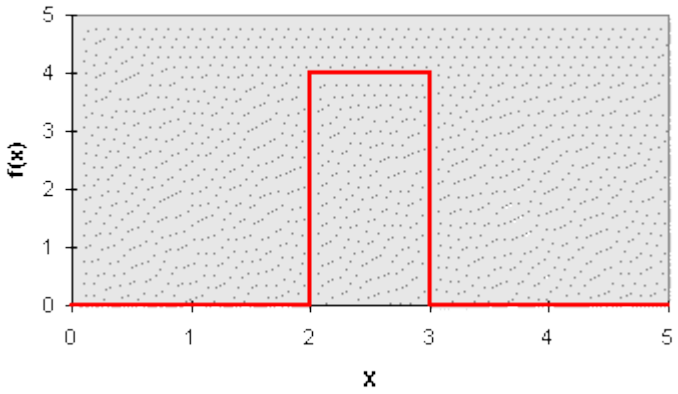
For example:



Although this function is a single function of x, we can force it into the multi-function format using:

$$\tilde{\pi}_i(x) = \begin{cases} f_i \text{ for } x_{i-1} < x < x_i \\ 0, \text{ for all other x} \end{cases}$$

This formally re-formulates the histogram as a sum of single-step functions, for example, $\tilde{\pi}_3 \quad x \quad$ is given by:



Using the probability mixing method, we will pick one of these steps using:

$$\tilde{\pi}_n = f_n \cdot (x_2 - x_1)$$

[NOTE: The probability of a step is NOT its height, but its area.]

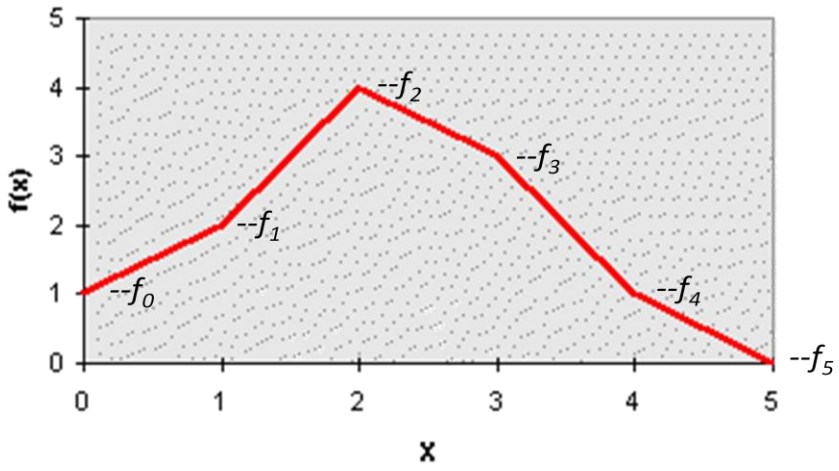
which can be normalized and used to choose j, which in this case is a choice of the jth "step".

Once a step has been chosen, the choice of a point on the step is done using:

$$x = x_{j-1} + (x_j - x_{j-1})\xi$$

Example: Piece-wise linear fits to continuous functions

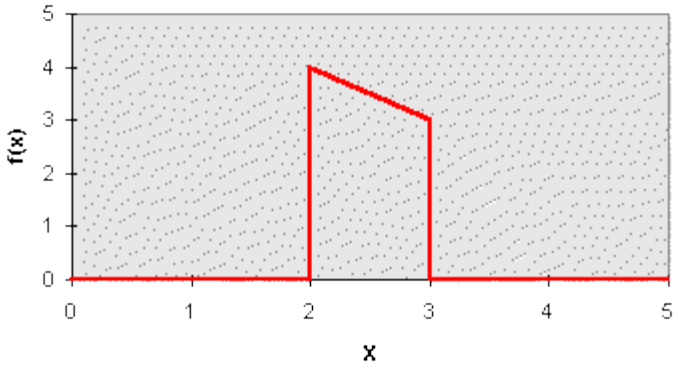
The same basic idea applies to linear continuous fits to continuous functions, which are like histograms except that a function is approximated with connected line segments:



Once again, we make this a sum of functions that are 0 except within a single region. If we let $f_0, f_1, \ldots, f_N$ be the values of the functions at the endpoints, the equations for the function within region i is:

$$\tilde{\pi}_i(x) = \begin{cases} f_{i-1} + \dfrac{x - x_{i-1}}{x_i - x_{i-1}} \cdot (f_i - f_{i-1}), & \text{if } x_{i-1} < x < x_i \\ 0, & \text{for other x} \end{cases}$$

For example, $\tilde{\pi}_3(x)$ is:



The choices that must be made are:

1. Choose a region *j* using the relative region probabilities of

$$\tilde{\pi}_i = \frac{f_{i-1} + f_i}{2} \cdot (x_i - x_{i-1})$$

2. Choose an x within region j using the relation:

$$x = x_{j-1} + (x_j - x_{j-1}) \left(\frac{-f_{j-1} + \sqrt{f_{j-1}^2 + f_j^2 - f_{j-1}^2 \ \xi_2}}{f_j - f_{j-1}} \right)$$

There is one more little "trick" of probability mixing that is not really recommended by mathematicians but is used in our Monte Carlo codes. You can use the same random number both to choose which distribution to use AND for choosing the value within it. (Mathematicians do not like this; I have heard it stated that "the later digits are not as random as the early digits". But, since we are going to use this trick later, I will describe it anyway.)

The basis of this is to form the CDFs for the first (discrete) choicee:

$$\Pi_i = \frac{\sum_{j=1}^{i-1} \tilde{\pi}_j}{\sum_{j=1}^{J} \tilde{\pi}_j} \qquad (2\text{-}11)$$

Then the first choice is *j* if the first random number falls within its domain, i.e., the chosen *j* is the one for which:

$$\Pi_{j-1} < \xi_1 < \Pi_j \tag{2-12}$$

The trick is to just let:

$$\xi_2 = \frac{\xi_1 - \Pi_{j-1}}{\Pi_j - \Pi_{j-1}} \tag{2-13}$$

This is nothing more than saying that the second random number is chosen to be the fractional position of the first random number within the chosen domain. Mathematically, this is sound, but you have to be aware that, for very thin domains, you might not have enough significant digits "left over" to get a good distribution within the domain. (But, on the other hand, who cares about getting good distributions within thin regions? It won't make much different in the result.)

2.4 Metropolis method

The Metropolis method is for choosing a stream of samples that will asymptotically conform to a desired distribution, but is subject to correlation effects. The main problem with using it is to know just when the random numbers it delivers can be trusted to be distributed like we want. This usually involves "warming it up" by throwing away the first *N* values. (The problem is, of course, figuring out N. It is said that it was devised at a three-couple dinner party in Los Alamos in the lat 1940s)

Nevertheless, despite this fact (and the fact that it hasn't seen much use in traditional nuclear engineering applications), it is an important technique in general Monte Carlo. (Plus, it is so different that it is quite interesting.)

Like rejection methods, the Metropolis method is very flexible for distributions which are not very well characterized, are not integrable, or are not invertable. In fact, the main requirement is that, for any given sample, its relative probability can be determined.

The basic procedure that I am going to present here is not as general as Metropolis can be (e.g., in the improved Metropolis-Hastings algorithm), but it will serve to illustrate the technique:

Step 1. Choose the first value of the variable uniformly in the domain (which implies a finite domain).
Step 2. Compute the relative probability of the current value.
Step 3. Tentatively choose a new variable, again uniformly in the domain.
Step 4. Compute the relative probability of the new variable.
Step 5. Make the tentative value the new choice:
  a. Unconditionally, if the relative probability of the new value is greater than the relative probability of the current value.

b. Otherwise with a probability equal to the ratio of the new relative probability to the current variable relative probability.

Step 6. Use the chosen value to determine the estimate of the effect of interest. (If you did NOT change to the new variable in Step 5, you RE-USE the same variable that you used in the step before.)

Step 7. Return to Step 2.

Actually, since the method is only guaranteed to approach the desired distribution asymptotically, it is usually a good practice to skip Step 6 for a few dozen cycles, to let the distribution settle in a bit.

2.5 Stratified sampling

Stratified sampling is another one of those techniques you need in your bag of tricks even though it is not used much in transport codes. It holds great promise, but I have found it to be surprisingly less useful than it appears to be at first glance. This is probably because (again) the technique is best applied to lower-dimensional problems. But you should know about it.

Conceptually, the technique attacks the discrepancy problem discussed in Chapter 1, i.e., trying to reduce the size of the "gaps" in the sampled space.

The idea is to provide some order to the random numbers used in a Monte Carlo simulation without going to the fully ordered quasi-random technique.

It is best shown with a simple example.

Let us return to an earlier problem and consider the mean of a uniform distribution between 0 and 1. As we saw in a previous section, the mean and standard deviation of this distribution is 0.5 ± 0.288675 .

The reason that I chose this example is because, with a flat distribution, the first term of the Koskma-Hlawka inequality is a constant, so the above uncertainty is entirely due to the discrepancy of the random numbers used. So, if we use, say, 100 samples to estimate the mean, we would expect a standard deviation of about 0.0289. Let's see if we can improve on this.

Let's do a divide-and-conquer approach and break this problem into two parts: Find the average between 0 and 0.5, find the average between 0.5 and 1.0, and average the two results. Since we have split the problem into two equal problems with ½ the domain, it won't surprise us that the resulting answers and uncertainty for each of these are:

$$0.25 \pm 0.144338 \text{ and } 0.75 \pm 0.144338$$

If we put them together, using the variance additive rules, the resulting guess becomes:

$$0.5 \pm 0.204124$$

which represents a factor of 2 reduction in the variance. We used exactly the same 100 ξ's (although we translated them into different sample points), and got half the variance.

Instead of literally running two problems, though, this improvement in efficiency can be obtained through the random number generator by forcing the LCG to stratify. This can be accomplished by letting the LCG do its regular thing to produce ξ_i and then change the result with a cyclical "stratum" counter:

$$\tilde{\xi}_i = (i + \xi_i - 1)/I \quad , i = 1, 2, ..., I \tag{2-14}$$

This will ONLY work in one dimension (i.e., in the Monte Carlo sense: each random deviate gives you one estimate of the answer), where the ξ's are all used to make the same decision. In multiple dimensions it is harder to implement.

Additional Observations:

1. The statistical formulas that we developed early in the course were developed for an unstratified random number generator. So, the standard deviations printed by our code from those formulas will not reflect the fact that these results are more accurate than pseudo-random results. If you stratify the LCG, you need to add a calculation of the true error to your printed results so you will be able to gauge the improvement.
2. In the limit, as the number of strata equals the number of random numbers drawn, you get an equal subdivision of each axis, which is a low-discrepancy set. Therefore stratified sampling conceptually stands as a compromise between pseudo-random Monte Carlo and quasi-random Monte Carlo.

Chapter 2 Exercises

Develop, code, and test direct inversion algorithms for choosing from the following distributions using direct inversion:

2-1. $\pi(x) = \sin x,\ 0<x<\pi$

2-2. $\pi(x) = 1 + x,\ 1<x<2$

Develop, code, and test algorithms for choosing from the following distributions using rejection:

2-3. $\pi(x) = \sin x,\ 0<x<\pi$

2-4. $\pi(x) = \begin{cases} x^2 & ,0<x<1 \\ e^{-2x} & ,1<x<2 \end{cases}$

Develop, code, and test algorithms for choosing from the following distributions using probability mixing:

2-5. $\pi(x) = 2x + \sin x,\ 0<x<\pi$

2-6. $\pi(x) = \begin{cases} x^2 & ,0<x<1 \\ e^{-2x} & ,1<x<2 \end{cases}$.

Develop, code, and test algorithms for choosing from the following distributions using the Metropolis method:

2-7. $\pi(x) = 2x + \sin x,\ 0<x<\pi$

2-8. $\pi(x) = \begin{cases} x^4 & ,0<x<1 \\ (2-x)^4 & ,1<x<2 \end{cases}$

2-9. Estimate the probability and the standard deviation of your answer that the sum of two random deviates (i.e., uniformly distributed real numbers between 0 and 1) is greater than 1.4.

2-10. Repeat the pi problem from Chapter 1, stratifying the choice of x into 10 strata. Compare to the standard deviation of the original algorithm.

2-11. The World Series involves a Best-of-Seven tournament between two teams, i.e., once a team has won four games, the contest is over. The locations of the games alternates between the two teams: 2 games in one place, 3 in the second, then 2 in the original place.

Is this fair? Assuming each team has a 50% chance of winning, write a MC code to determine the expected number of games played in each location.

2-12. A commonly used algorithm to select normally distributed variables is:

1. Pick x=-ln(ξ_1)
2. Pick y=-ln(ξ_2)
3. Keep x if y>$(x-1)^2/2$; otherwise, repeat 1&2.
4. x is made negative if $\xi_3<0.5$

Demonstrate that this rejection method has an expected distribution of *x* proportional to $e^{-x^2/2}$ with efficiency of about 76%. (Use a normal variable distribution table to compare your results.)

2-13. Explain how the algorithm in the previous problem has an explicit value proportional to $e^{-x^2/2}$. (That is, find the product of the probability that x is chosen times the probability it is kept.)

Answers to selected exercises

Chapter 2

2-1. $x = \cos^{-1}(1-2\xi)$

2-2. $x = \sqrt{4+5\xi} - 1$

2-3. 1. $x = \pi\xi_1$

2. Keep x IFF $\xi_2 < \sin(x)$

2-4. 1. $x = 2\xi_1$

2. Keep x IFF EITHER $x < 1$ *AND* $\xi_2 < x^2$ OR $x > 1$ *AND* $\xi_2 < e^{-2x}$

2-5. *IF* $\xi_1 < 0.831502$, *use* $x = \pi\sqrt{\xi_2}$, *OTHERWISE use* $x = \cos^{-1}\ 2\xi_2 - 1$

2-6. *IF* $\xi_1 < 0.850681$ *use* $x = \sqrt[3]{\xi_2}$, *OTHERWISE use* $x = \dfrac{-\ln\ e^{-2} - \xi_2\ \ e^{-2} - e^{-4)}}{2}$

2-9. $x = 0.18 \pm \dfrac{0.384}{\sqrt{N}}$

2-11. Team A: $2.94 \pm \dfrac{0.827}{\sqrt{N}}$

Team B: $2.88 \pm \dfrac{0.330}{\sqrt{N}}$