

Statistics

Lecture 5

August 31, 2000

Frank Porter

Caltech

The plan for these lectures:

1. The Fundamentals; Point Estimation
2. Maximum Likelihood, Least Squares and All That
3. What is a Confidence Interval?
4. Interval Estimation
5. Monte Carlo Methods

Additional topics covered by Roger Barlow and Norman Graf

Monte Carlo – Introduction

The Monte Carlo method is a numerical technique, which, in general terms, is directed at the problem of computing integrals.

- It is especially handy in multi-dimensional problems.
- The “simulation” aspect is very useful for us – we can generate a dataset of Monte Carlo “events”, and decide later what integral(s) we wish to evaluate.

Monte Carlo Integration

Suppose we wish to evaluate the integral:

$$I = \left(\int \cdots \int \right)_{\{\mathbf{x}\}} f(\mathbf{x}) d^k \mathbf{x},$$

over a k -dimensional unit cube $\{\mathbf{x}\}$.

We can re-express this integral in the form of an expectation value with respect to a probability distribution, $p(\mathbf{x})$ which is uniform over $\{\mathbf{x}\}$, and zero elsewhere:

$$\begin{aligned} I &= \left(\int \cdots \int \right)_{\{\mathbf{x}\}} f(\mathbf{x}) p(\mathbf{x}) d^k \mathbf{x} \\ &= \langle f \rangle. \end{aligned}$$

MC Integration (Continued)

Since the sample mean is an unbiased estimator for the expectation value, a plausible estimate for the integral can thus be obtained with:

$$I_N = \frac{1}{N} \sum_{i=1}^N f(\mathbf{x}_i),$$

where \mathbf{x}_i are N vectors drawn from $p(\mathbf{x})$, i.e., uniformly sampled in the unit cube.

Exercise: Modify the discussion for an arbitrary finite region. What about infinite regions? What assumptions are we making?

MC Integration (Continued)

Clearly,

$$I = \langle I_N \rangle. \quad (\text{Unbiased})$$

$$I = \lim_{N \rightarrow \infty} I_N. \quad (\text{Consistent})$$

Also, (You show):

$$\text{Var}(I_N) = \frac{1}{N} \text{Var}(f)$$

We haven't accomplished much yet – why not simply divide our space up into N equally-spaced points and average the function values over these points?

Depending on the function, this could be biased. But at least it is consistent.

However, instead of sampling uniformly, we can adjust our sampling to save computer time, by concentrating function evaluations where f is largest.

MC Simulation, Random Number Generation

In HEP, useful concept is simulation of collision events and response of detector to particles. Such a simulation is typically built from several distinct random processes – The problem can be broken down into smaller simulation tasks.

At the heart of any simulation is the underlying differential distribution, which we can generally relate to a probability distribution over some set of possible outcomes.

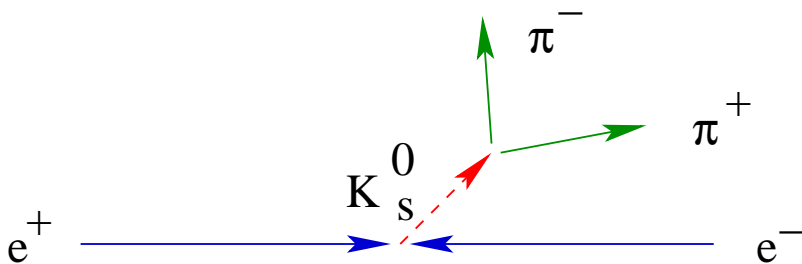
Thus, the task of simulation reduces to the problem of generating numbers as if they were sampled from a specified pdf. These are our “**Random Numbers**”.

We now discuss methods of generating these random numbers.

Sample Problem

It may be helpful to have in our minds a sample problem which is more-or-less a realistic example of the sort of thing we wish to do.

Thus, suppose we are interested in the efficiency for detecting K^0 's produced in e^+e^- collisions according to a $1 + \cos^2 \theta$ distribution, and a momentum distribution falling as p^{-n} . We are given a detector which can track charged particles in the central 90% of the solid angle. An analysis cut is placed on decay vertices, accepting those within the region (1, 10) cm from the collision point. We assume the measurement of the tracks are normal in curvature, azimuth, and $\tan(\text{dip})$ with no correlations in these variables.



Towards solving the Sample Problem

1. First, generate the “event”:

- a) In principle, can start with generating whether we have a K_S^0 , and whether it decays to $\pi^+\pi^-$, but often we just generate only $K_S^0 \rightarrow \pi^+\pi^-$, and correct for missed modes explicitly at the end.
- b) Generate the angle of the K (azimuth doesn't matter, in this simple problem), and its momentum.
- c) Generate the decay $K \rightarrow \pi^+\pi^-$, isotropic in the kaon frame, and Lorentz transform to lab.
- d) Generate the decay length of the kaon, according to its momentum and the exponential decay law.

Sample Problem (continued)

2. Second, generate the “detector response”:
 - a) Cut event if the vertex is not in fiducial volume.
 - b) Cut event if either track is outside fiducial volume.
 - c) Generate measured momenta, directions.
3. Finally, perform final “analysis”, e.g.:
 - a) Cut event if mass is “inconsistent” with K .

Finally, obtain the efficiency as the ratio of events surviving to the end, divided by the number generated.

Inverse Transform Method

Consider a 1-dimensional problem in which we wish to “simulate” a process with pdf $f(x)$, $x \in (a, b)$. We are given a “pseudo-random number generator” which is uniform:

$$p(u) = \begin{cases} 1 & u \in (0, 1) \\ 0 & \text{otherwise.} \end{cases}$$

Want to find a transformation $u \rightarrow x$ such that x is drawn from $f(x)$.

Let $x = T(u)$ be that transformation. Must have:

$$f(x)dx = f[T(u)] \left| \frac{dx}{du} \right| du = p(u)du.$$

Therefore:

$$f[T(u)] \left| \frac{dx}{du} \right| = p(u).$$

Hence, using $x = T(u)$:

$$f[T(u)] |dT(u)| = p(u)du.$$

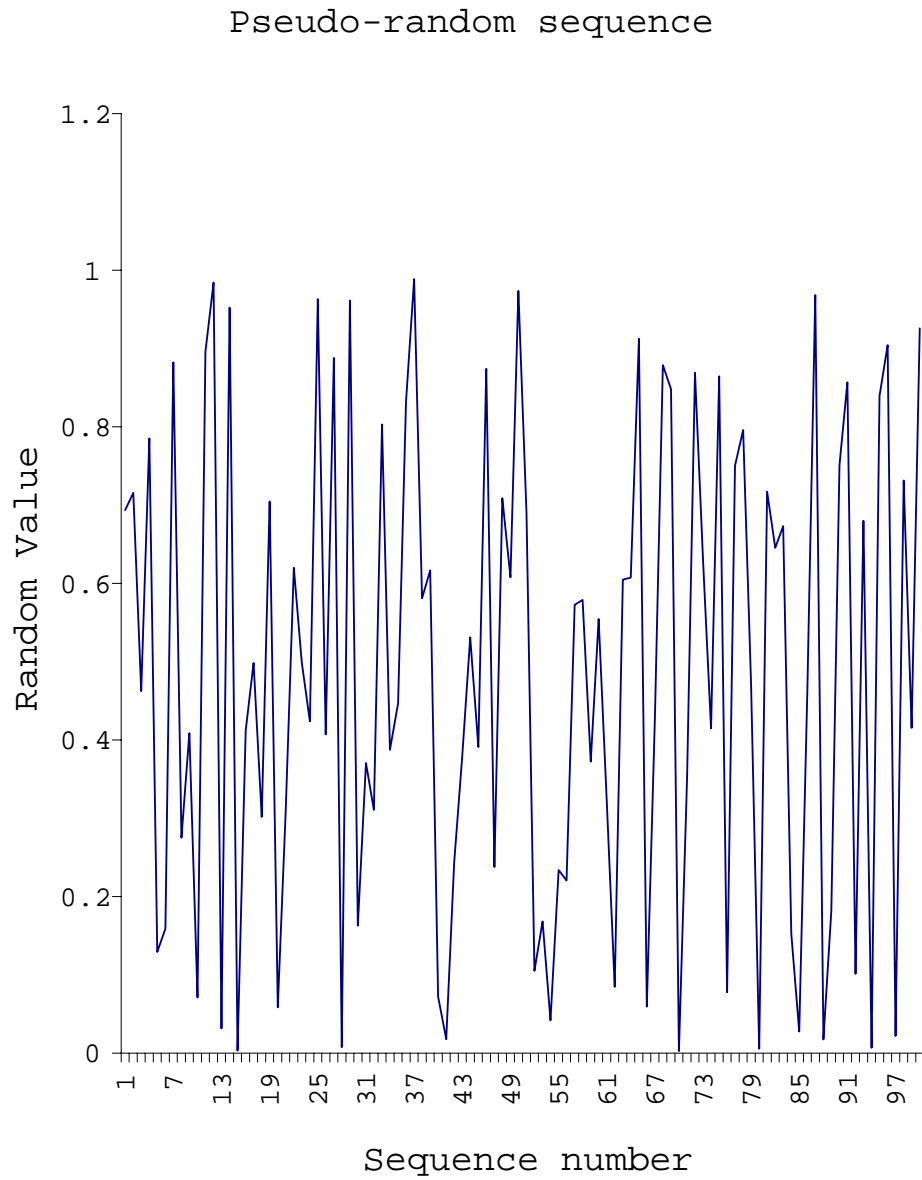
Aside: Uniform random number generation

There is a vast literature on the generation of uniformly-distributed (pseudo)-random numbers. In computer simulations, we nearly always make use of pseudo-random sequences, which are deterministic, yet have many properties as if they were truly random.

To get the flavor, here is a very simple such generator. It repeats after 2048 numbers. (Based on an old “VM module 18” note!)

```
float rand() {
    static int k=111;
    /* assume mod takes modulus */
    /* wrt 2nd argument */
    k=mod(5*mod(5*mod(5*k,8192),8192),8192);
    return(k/8192.);
}
```

Sample Pseudo-Random Sequence



Inverse Transform Method (continued)

If $T(0) = a$, then $dT/du \geq 0$, and we integrate:

$$\int_{T(0)}^{T(u)} f[T(u')] dT(u') = \int_0^u p(u') du' = u.$$

Let

$$F(x) = \int_a^x f(x') dx'.$$

Then $F[T(u)] = u$, or

$$x = T(u) = F^{-1}(u).$$

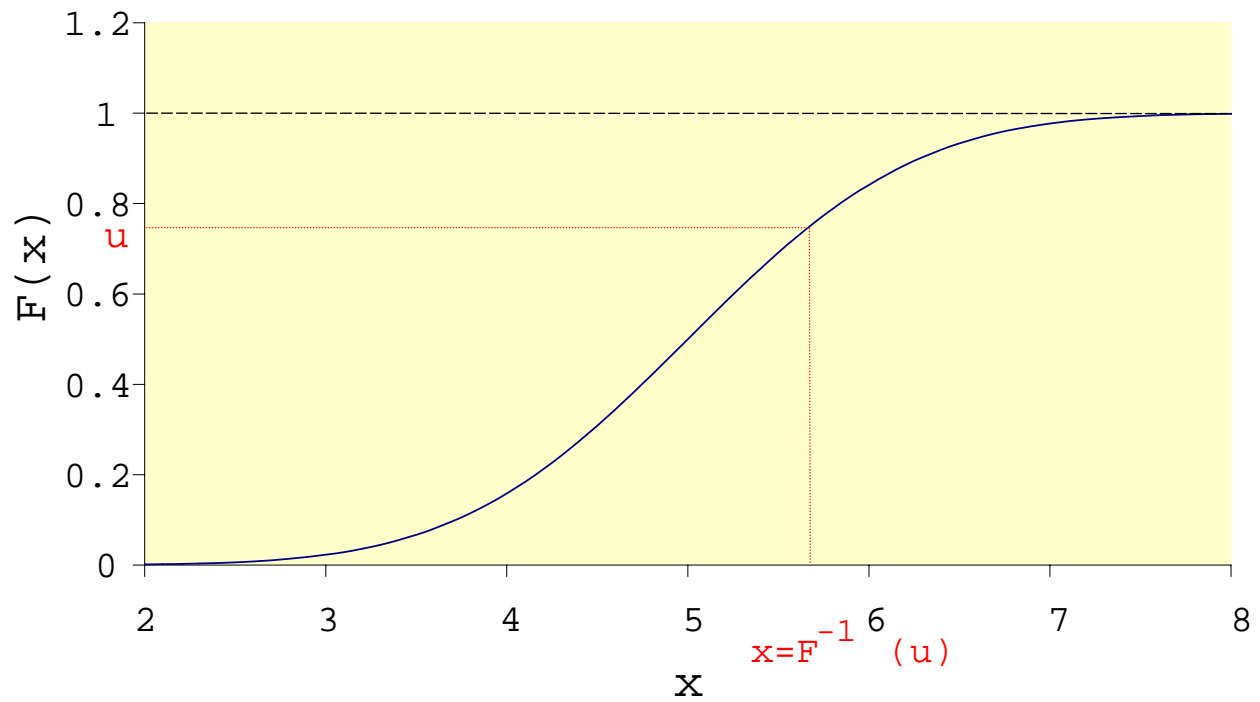
That is, we solve for x in:

$$\int_a^x f(x') dx' = u.$$

Makes sense: We sample most those regions where $F(x)$ is changing rapidly, i.e., where $f(x)$ is largest.

Illustration of Inverse Transform Method

Inverse Transform Method



Inverse Transform – Example

Suppose we wish to generate random numbers according to the $N(0, 1)$ distribution, using our source of uniform random numbers, u , on $(0, 1)$.

The prescription says to solve for x in:

$$u = \int_{-\infty}^x \frac{1}{\sqrt{2\pi}} e^{-x'^2/2} dx'.$$

Hmm... Not so easy. But recall trick of evaluating “square”, and transforming to polar coordinates:

$$\begin{aligned} p(x, y) dx dy &= f(x) f(y) dx dy \\ &= r e^{-r^2/2} dr \frac{d\phi}{2\pi} \\ &= g(r) h(\phi) dr d\phi. \end{aligned}$$

Thus, generate r and ϕ according to $g(r)$ and $h(\phi)$, and transform the result to x and y to obtain two normally distributed random numbers.

Inverse Transform – Exercises

Exercise: Complete the normal example above.

Exercise: Apply the inverse transform method to the “linear” pdf: $f(x) = |x|$ for $x \in (-1, 1)$.

Exercise: Apply the inverse transform method to the Cauchy (Breit-Wigner) distribution.

Discrete Distributions

The inverse transform method may be applied to discrete distributions also:

Let $p(n), n = 0, 1, 2, \dots$ be a discrete probability distribution. We may sample from this distribution, given a uniform (on $(0, 1]$) sampling u by solving for i in:

$$\sum_{n=0}^{i-1} p(n) < u \leq \sum_{n=0}^i p(n).$$

Exercise: Write a routine to generate Poisson-distributed numbers according to this algorithm.

Poisson Example

But, as for continuous distributions, for given distributions, there may be other tricks:

Exercise: Make sense of the following C-code fragment as a Poisson number generator:

```
int poisson(double mean) {
/* Assume rand() is a source of uniform */
/* random numbers on (0,1]                */
  double sum, rand();
  int n;
  for(n=0, sum=-log(rand()));
    /* terminates for loop when false */
    sum<mean;
    sum-=log(rand()), n++) {}
  return(n);
}
```

Composition Method

If we have a distribution we wish to generate that is a sum of terms, it may be easier, or save computation time, if we break it up into pieces.

For example, suppose we can decompose the desired pdf $f(x)$ as:

$$f(x) = r f_a(x) + (1 - r) f_b(x),$$

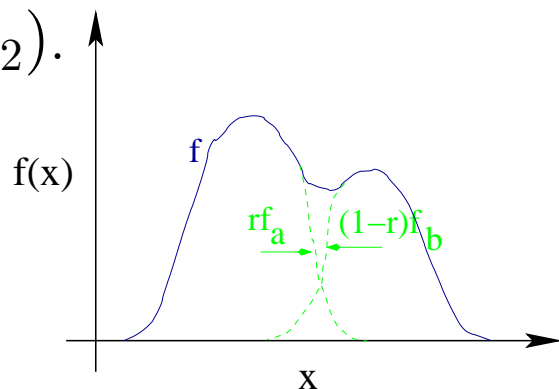
where f_a and f_b are themselves normalized pdfs ($0 \leq r \leq 1$).

Then we can generate a sampling from f by:

1) Generate two uniform randoms u_1 and u_2 .

2) If $u_1 < r$ let $x = F_a^{-1}(u_2)$.

If $u_1 \geq r$ let $x = F_b^{-1}(u_2)$.



Composition Method - Example

Suppose we wish to generate a $1 + \cos^2 \theta$ angular distribution on $0 \leq \theta \leq \pi$.

We may break this up into the sum of two distributions:

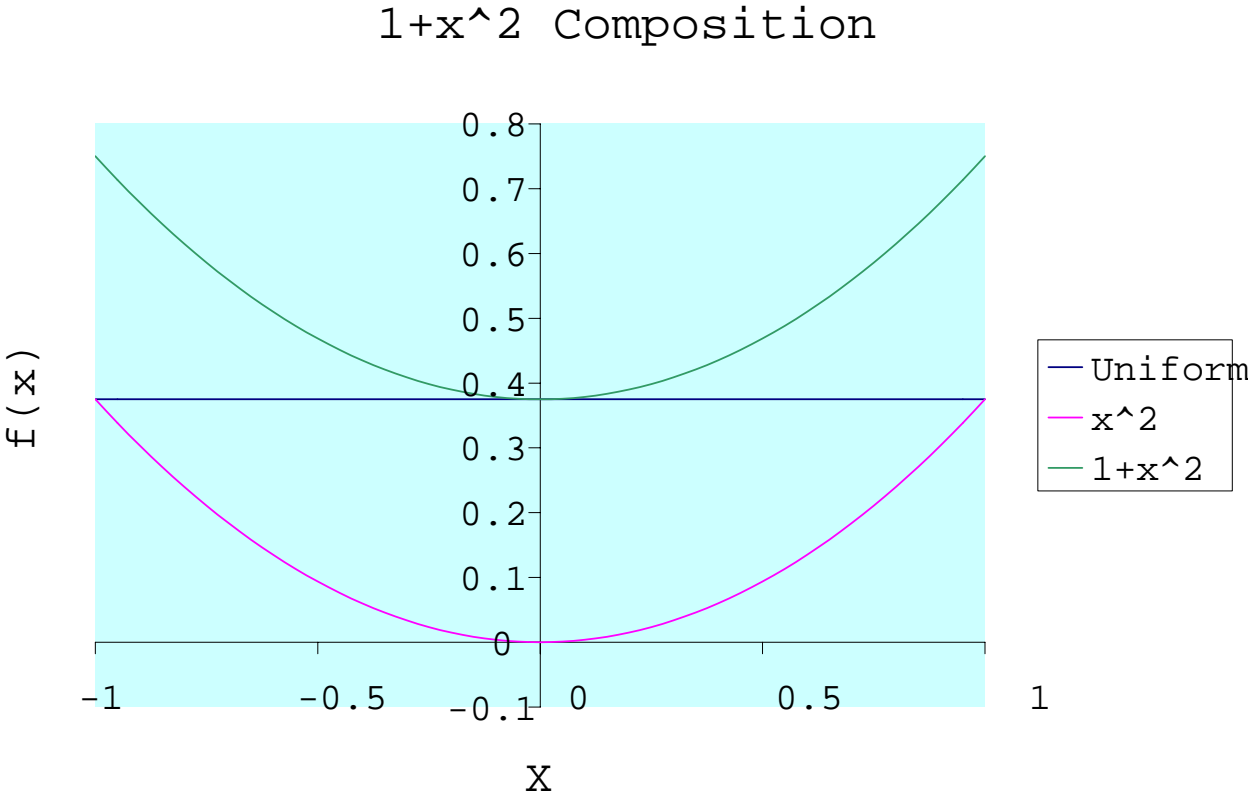
$$f(x = \cos \theta) = \frac{3}{4} \cdot \frac{1}{2} + \frac{1}{4} \cdot \frac{3}{2} x^2.$$

The resulting algorithm is:

- 1) Generate uniform randoms u_1 and u_2 .
- 2) If $u_1 < 3/4$, let $x = 2u_2 - 1$.
- 2') Otherwise, let $x = (2u_2 - 1)^{1/3}$.

Exercise: Do the above angular distribution by the (brute force) inverse transform, and contrast your result with the composition method.

Illustration of Composition Method Exercise

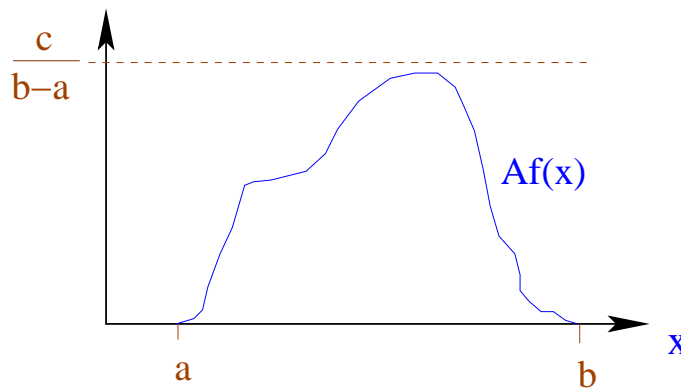


Acceptance-Rejection Method

It may be that computing the inverse transform, even on a decomposed distribution, is intractable. A powerful (but potentially inefficient) method is the **acceptance-rejection method**. Here, the algorithm is:

- 1) Find numbers a, b such that $f(x) = 0$ whenever $x \notin (a, b)$. (Best to find largest such a and smallest such b).
- 2) Find maximum of $Af(x)$, or at least some number greater than the maximum, and define a number c such that:

$$\frac{c}{b-a} \geq Af(x), \quad \forall x$$

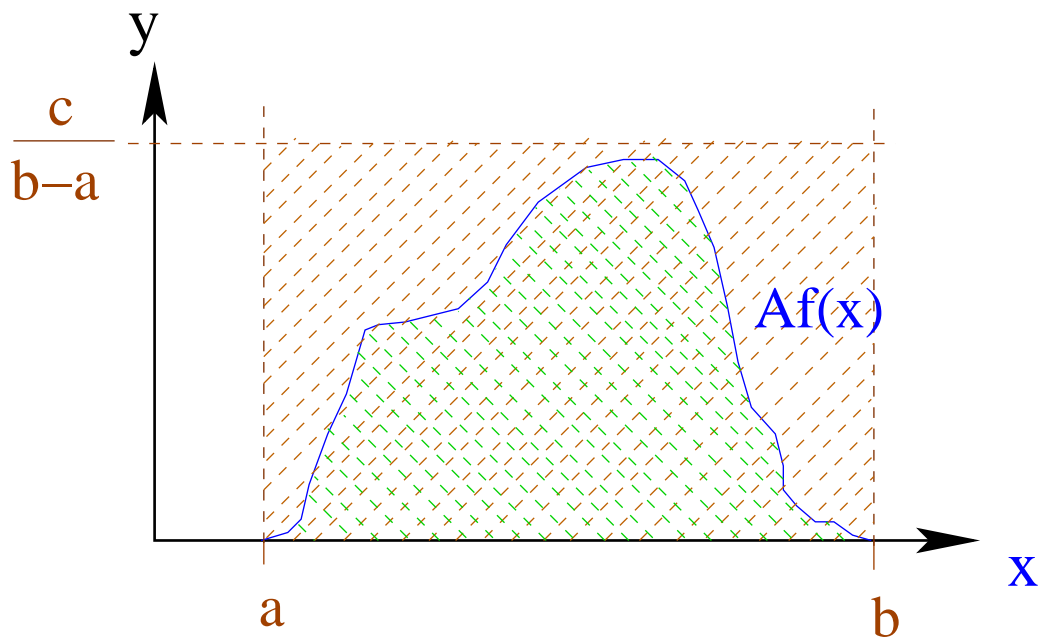


Acceptance-Rejection Method (continued)

- 3) Sample an x uniformly on (a, b) .
- 4) Sample a y uniformly on $(0, c/(b - a))$.

Note that the (x, y) pairs so generated populate a box uniformly.

- 5) Evaluate $Af(x)$.
 - (a) If $y \leq Af(x)$, accept x .
 - (r) If $y > Af(x)$, reject x , go back to step 3.



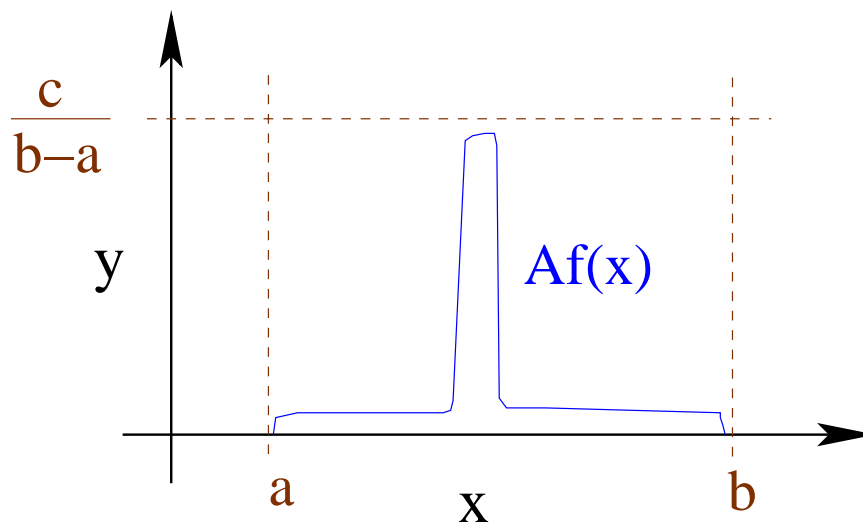
Acceptance-Rejection – Discussion

Notice why this works: From the set of points (x, y) uniformly distributed over the box, we accept only those which lie under the $Af(x)$ curve.

The fraction of trial points accepted is:

$$\begin{aligned}\epsilon &= \frac{\text{Area under } Af}{\text{Area of box}} \\ &= \frac{A}{c}.\end{aligned}$$

While this method is very easy to apply, it may be very inefficient, even if we work hard to make the box as small as possible.



Importance Sampling

At the cost of complexity, we may mitigate the efficiency problem in acceptance-rejection sampling by modifying the simple box distribution to one which more nearly approximates the desired distribution.

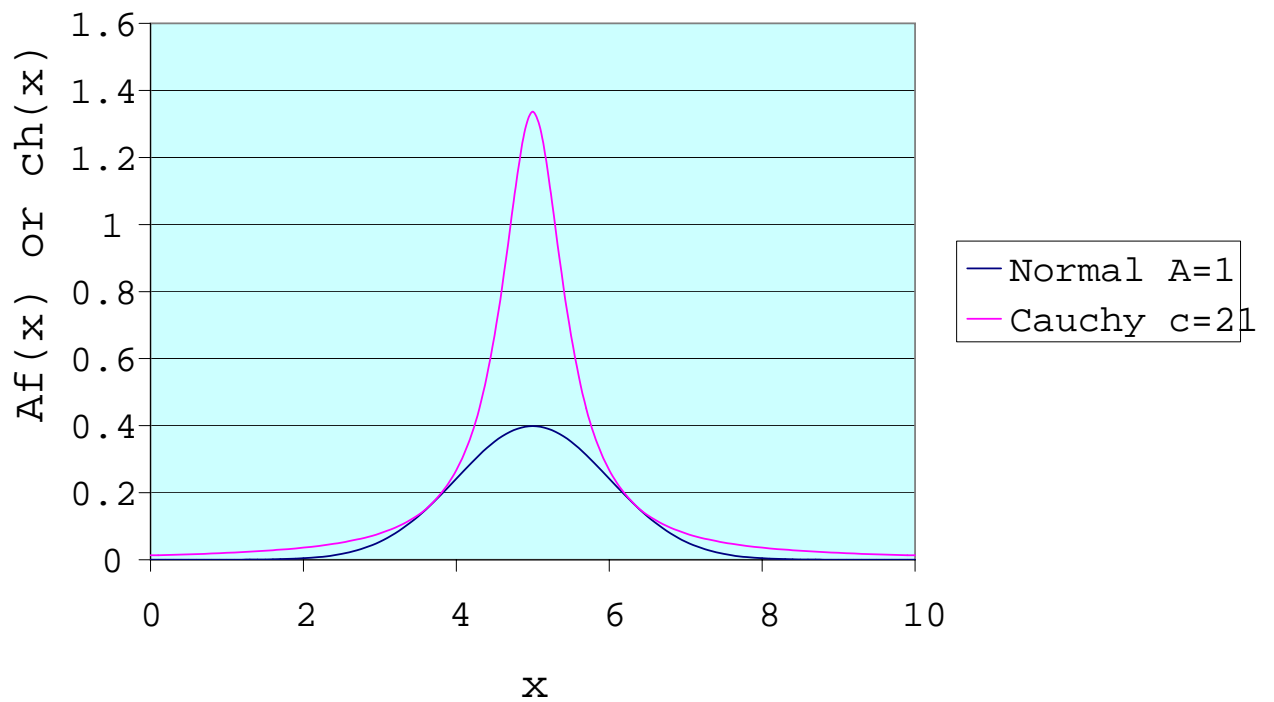
Of course, whatever shape we approximate the distribution with should be tractable in terms of sampling.

- 1) Thus, given the problem of sampling from $f(x)$, we find some suitable pdf $h(x)$ such that $ch(x) \geq Af(x)$ for all x ($c > 1$ if $A = 1$).
- 2) Choose an x according to $h(x)$, and a y uniform on $(0, ch(x))$.
- 3) If $y > Af(x)$, reject x and try again, otherwise, accept x .

This is the method of “**Importance Sampling**”.

Example of Importance Sampling

Importance Sampling -- Normal with Cauchy



Von Neumann Theorem

Theorem: (Von Neumann) Represent pdf $f(x)$ as:

$$f(x) = Cg(x)h(x),$$

where $C \geq 1$ is a constant, $h(x)$ is a pdf (chosen for convenience of generation of random numbers according to $h(x)$), and $g(x)$ is the “correction function” relating $Ch(x)$ and $f(x)$, with the constraint $Ch(x) \geq f(x)$.

Sample a value y uniformly on $(0, 1)$, and an x according to $h(x)$.

Then the distribution of x , under the requirement $y \leq g(x)$, is just $f(x)$.

The “**efficiency**” (fraction of trials accepted) is $1/C$.

Proof: Exercise. **Hint:** You want to show that $P(x|y \leq g(x)) = f(x)$. Use Bayes’ theorem.

Distribution of Number of Trials

The distribution of n , the number of failures before a successful trial is simply (letting ϵ be the efficiency):

$$p(n) = (1 - \epsilon)^n \epsilon, \quad n = 0, 1, 2, \dots$$

This is the “geometric distribution”.

