Wintersemester 2011/2012, ESS

Übung: Generating Gaussian Pseudorandom Numbers

Assignment due date: 17.12.2011
Exercise on 12.12.2011

One frequently requires a source of random numbers with a certain distribution. In this exercise, you will implement different algorithms for generating random numbers with a Gaussian (or Normal) distribution from uniformly distributed random numbers. At least one of the functions that you write will serve as the basis for the next exercise (simulating error propagation). The probability density function for a Gaussian distribution with mean μ and variance σ^2 is

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

You can use any programming language you like, as long as your program is compilable and runnable on the machines in the student pool. Your program should read the method to use, the number of random numbers to print out, and a seed value to initialise the random number generator from the command-line. If you call your program `gauss`, it should take the following command-line parameters:

```
./gauss  <method to use>  <number of random numbers to print>  <seed value>
```

Here is an example run that prints 3 random numbers using the Box-Muller transform and a seed value of 42:

```
$ ./gauss box-muller 3 42
 0.289357
-1.591773
-1.076735
```

In general it is important that a program run is repeatable, so make sure your program always produces the same output for a given seed value.

If you are programming in C, you can use the `drand48` function to get a uniformly distributed double precision floating point number in the interval $[0.0, 1.0)$. You can seed the random number generator with the `srand48` function. Some maths functions such as `sin`, `cos`, or `log`, require you to link to the standard math library. You can do this when linking/compiling with `gcc` by adding `-lm` to the command-line like so:

```
gcc -Wall -Wextra -O2 foo.c -lm -o foo
```

Your assignment

Please submit a brief report with your answers to ast_uebung[at]zbh[dot]uni-hamburg[dot]de (please include your full name). Attach one file containing the answers to all assignments (only .txt or .pdf), and one file containing the source code for your program.

Plotting Histograms with R

There is a small script written in the programming language R located in `/home/matthies/uebung-rng/plot.r`. Copy it to the directory you are working in. If you have a file `foo.dat` with one number on each line you can get a histogram of the numbers by running `./plot.r foo.dat`.

Save the output for each algorithm to a different file:

```
./gauss box-muller 1000 42 > bm.dat
./gauss accept-reject 1000 42 > ar.dat
./gauss sum-uniform 1000 42 > su.dat
```

You can then compare all three plots with

```
./plot.r bm.dat ar.dat su.dat
```

This will produce a PDF file `Rplots.pdf` in the current directory.

Aufgabe 1: Box-Muller transform

Let u_1, u_2 be independent uniformly distributed random variables with $u_1, u_2 \in (0, 1]$. Then

$$g_1 = \sqrt{-2\ln(u_1)}\cos(2\pi u_2)$$

$$g_2 = \sqrt{-2\ln(u_1)}\sin(2\pi u_2)$$

are two random numbers from a Gaussian distribution with mean 0 and variance 1.

Implement and test the transform. Generate 5000 random numbers and check the distribution using the histogram plotter given above. Include the plot in your report. Explain why the random numbers have to be in the range $(0, 1]$ and not $[0, 1)$ and how you implemented this in your program.

Bonus question: what are (approximately) the smallest and largest numerical values produced by this random number generator in single- and double-precision floating point arithmetic?

Aufgabe 2: Acceptance-rejection sampling

Acceptance-Rejection sampling is a general method that is useful when we either don't have a direct transform (e.g. the Box-Muller transform above) available or when it is too time-consuming to compute. The idea is to sample from a different probability density function g that can enclose the probability density function f we are interested in. If we reject samples with the right probability the samples that are accepted will be distributed according to f. When we reject a sample, we have to repeat the whole process until we find a sample that we accept.

We accept samples x with a probability $f(x)/Mg(x)$, where $M > 1$ is a bound on $f(x)/g(x)$ so that $f(x) < Mg(x)$.

Note that the range of possible values for both distributions has to be the same. Whereas the uniform distribution has a finite range, the normal distribution takes on values from $(-\infty, \infty)$, so we cheat a little bit and approximate the normal distribution on the interval $[-10, 10]$, i.e. with its tails cut off.

Here, use uniformly distributed random numbers in the range $[-10, 10]$ for the distribution g and choose $M = 1/g(x)$ ($g(x)$ is constant) so that the acceptance probability becomes $f(x)$. To generate a sample from $f(x)$, draw numbers from the uniform distribution until a sample is accepted.

Implement and the algorithm and test it the same way as the Box-Muller transform. How many samples must be drawn from the uniform distribution on average for each normally distributed sample that is generated? You can either derive this analytically or measure it empirically by using two counters in your program, one for the number of uniform random numbers generated and one for the number of accepted samples.

Aufgabe 3: Summing uniform random numbers

The central limit theorem tells us that the sum of independent identically distributed random variables converges towards the normal distribution as the number of terms in the sum increases. We can therefore take the sum of n uniformly distributed random variables $s = u_1 + u_2 + ... + u_n$ as an approximation of the normal distribution. We must shift and rescale the numbers drawn from this approximate distribution to get the desired mean $\mu = 0$ and variance $\sigma^2 = 1$ of the standard normal distribution. This can be achieved by subtracting the mean $\mu_s = \frac{n}{2}$ from s and then dividing by its standard deviation $\sigma_s = \sqrt{\frac{n}{12}}$.

Let $s = u_1 + u_2 + ... + u_n$. Then the distribution of

$$g = \sqrt{\frac{12}{n}}\left(s - \frac{n}{2}\right)$$

converges towards a normal distribution with mean 0 and variance 1 as n goes to infinity.

Implement the algorithm and test it the same way as the Box-Muller transform. Choose $n = 5$ for your implementation or try your own values. What are the smallest and largest numerical

values of the random numbers that can be produced by this method? Also, prove that s really has variance $\sigma_s^2 = \frac{n}{12}$ (hint: start by showing that a single random variable uniformly distributed on the range $[0, 1]$ has variance $\sigma^2 = \frac{1}{12}$).