Stat 400, Summer I June 25, 1996

Handout on Simulation

II. COMPUTER SIMULATION OF CONTINUOUS & DISCRETE R.V.'S

A. PSEUDORANDOM NUMBERS

The main idea of this topic is that there are simple and fast algorithms for generating sequences of numbers $U_1, U_2, \ldots$ on the computer which *'look random and uniformly distributed'* in the sense that for any numbers $0 < a < b < 1$, for large N

$$\frac{1}{N} \cdot \#\{j = 1, 2, \ldots, N \,:\, a < U_j < b\} \;\approx\; b - a$$

and also that there is no detectable pattern in the sequence which would allow you to answer probabilistic questions about later sequence elements U_{n+k}, $k \geq 1$ more precisely with knowledge of $U_1, U_2, \ldots, U_n$ than without. Such artificially, but deterministically, generated numbers are called **pseudorandom**, and standard routines generating them are often included in computer languages and packages (including BASIC, FORTRAN, PASCAL, Mathematica, and all Statistical packages like MINITAB, SAS, ...). Here are two types of algorithms used (but there are now several more, less well-established):

Algorithm 1. Linear-Congruential Generator.
Start with a large integer m like the word-size 2^{32}. All of the numbers X_k, $k = 1, 2, \ldots$ generated in sequence will be whole numbers ranging from 0 to $m - 1$, which will look equally likely to take on any of these values, and we can divide them all by m to get approximately uniformly distributed numbers $U_k = X_k/m$. To express the algorithm, fix integers a, b between 1 and $m - 1$. (It is important to choose these well: for example b and m should have no factors in common.) Next take the first number in the sequence X_0 (the *seed*) more or less arbitrarily between 0 and $m - 1$. (The clock-time expressed as an integer number of hundredths of seconds is one way to choose.) The further numbers in the sequence of X's follow the rule

$$X_k \;=\; \text{remainder after dividing} \quad a + b \cdot X_{k-1} \quad \text{by} \quad m$$

Two particular successful choices for b with $m = 2^{32}$ are:

$b = 69069$ (Marsaglia) $b = 1812433253$ (Borosh & Niederreiter)

With these choices, the choice of a does not matter much.

Algorithm 2. Additive Shift-register Generator.

Start again with the large integer m, usually a word-size, as above. Start also with an arbitrarily filled array $Z = (Z_1, \ldots, Z_{55})$ of 55 whole numbers between 0 and $m - 1$. (These could for example be obtained as the first 55 numbers output by algorithm 1.) The output of the sequence results from repeating the following pair of operations: next

$X_k = Z_{27} + Z_{55}$ *(and subtract* m *if the result is* m *or greater)*; then replace Z by the shifted sequence $(X_k, Z_1, Z_2, \ldots, Z_{54})$ with new first entry.

As in the previous algorithm, we produce a sequence of discrete-Uniform looking random numbers between 0 and $m-1$. To obtain uniform numbers between 0 and 1, just divide them by m.

The classic computer-science book of D. Knuth, **The Art of Computer Programming**, vol. 2 on *Seminumerical Algorithms* is slightly old but still the best reference on this topic, particularly as regards Algorithm 1, which is still the most used.

B. SIMULATING CONTINUOUSLY DISTRIBUTED RANDOM NUMBERS

We have already seen the idea here, in Example C of the Transformation of Random Variables handout. Suppose we want to simulate a *rv* Y with *cdf* $F_Y = G$. Then that Example showed $Y = g(X) = G^{-1}(X)$ *is* such a random variable, if $X \sim Unif[0,1]$. For example, if we wanted to simulate an $Exponential(\lambda)$ r.v., we solve the equation $x = G(y) = 1 - e^{-\lambda y}$ (since the latter is the $Expon(\lambda)$ *cdf*), to find $y = g(x) = G^{-1}(x) = -\frac{1}{\lambda}(1 - \log(x))$. Thus, if $X \sim Unif[0,1]$, then $Y = -\frac{1}{\lambda}(1 - \log(X)) \sim Expon(\lambda)$. Actually, as you can check using the methods (formula $(1')$) of the Transformation handout, if $X \sim Unif[0,1]$, then $1 - X \sim Unif[0,1]$, so that the random variable $Z \sim -\frac{1}{\lambda}\log(X)$ is also distributed as $Expon(\lambda)$.

Here is another example. Suppose you wanted to simulate a random variable Y with the not particularly elegant density

$$f(y) = 3(1+y)^{-4}\,, \quad y > 0$$

The method is first to find, and then to invert, the *cdf* F_Y, as follows. For $y > 0$,

$$F_Y(y) = \int_0^y \frac{3}{(1+t)^4}\,dt = \frac{-1}{(1+t)^3}\,\Big|_0^y = 1 - (1+y)^{-3}$$

Then, putting $x = F_Y(y)$, we solve for $y = g(x) = F_Y^{-1}(x)$:

$$g(x) = (1-x)^{-1/3} - 1$$

From this and our previous discussion, it follows that if X is a $Unif[0,1]$ (pseudorandom) number, then $Y = (1-X)^{-1/3} - 1$ (or $Z = X^{-1/3} - 1$) is a pseudorandom variable with *cdf* F_Y and density f as given.

C. SIMULATING DISCRETE RANDOM NUMBERS

Suppose we want to simulate (as always, from the starting point of a $Unif[0,1]$ *rv* X) a discrete *rv* Y which takes each of the values 0.1, 9, -3, 17 with respective probabilities 0.13, 0.45, 0.22, 0.20. The idea is very simple. Since X falls in any specified subinterval of $[0,1]$ with probability equal to the subinterval's length, just subdivide $[0,1)$ into the four non-overlapping intervals $[0, 0.13)$, $[0.13, 0.58)$, $[0.58, 0.80)$, $[0.80, 1)$. You

see that what this does is just to lay the four probabilities for Y values 'end-to-end' as lengths: the lengths have to cover the interval exactly because the probabilities must sum up exactly to 1. So a good rule for generating the *rv* Y is:

respectively if	$X \in [0, 0.13)\,,\ [0.13, 0.58)\,,\ [0.58, 0.80)\,,\ [0.80, 1)$
put	$Y = 0.1\,,\ 9\,,\ -3\,,\ 17$

It is easy to see that this discrete *rv* has precisely the prescribed values and probability mass function. The idea is easy to generalize to any discrete *pmf* with finitely many values, and even — after a little thought — to countable-valued discrete *rv*'s like the Poisson.

D. SIMULATING MORE COMPLICATED FUNCTIONS OF MORE (INDEP.) RV'S

One important reason for simulating at all is that we want to understand operationally how certain combinations or functions of *many* independent random variables behave, but that we cannot calculate the relevant probabilities easily. It is easy and quick to create on the computer many large batches of random variables with prescribed probability distributions. Therefore we simulate in order to answer interesting probability questions through **relative frequencies**. (*Remember the* **relative-frequency interpretation** *of probabilities ?*) Here is a simple example: suppose we want to verify the (true) fact that a sum of two indpendent $\mathcal{N}(0,1)$ *rv*'s has a $\mathcal{N}(0,2)$ probability distribution. This statement can be made into a prediction about (large numbers of pairs of) simulated variables, as follows: simulate 2000 independent $N(0,1)$ *rv*'s, and arrange them into pairs $(Z_{1,1},\, Z_{1,2})$, $(Z_{2,1},\, Z_{2,2})$, $\ldots$, $(Z_{1000,1},\, Z_{1000,2})$. Then our prediction is that, for any fixed number r, $\frac{1}{1000}$ times the number of pairs $(Z_{j,1},\, Z_{j,2})$ such that $Z_{j,1} + Z_{j,2} \le r$ will be aproximately $\Phi(r/\sqrt{2})$. Let us try this out and test it. I simulated just such a batch of *rv's* and, as a function of r plotted

$$\frac{1}{1000} \times \#\{j = 1, \ldots, 1000 \,:\, Z_{j,1} + Z_{j,2} \le r\} \qquad vs. \qquad \Phi\Big(\frac{r}{\sqrt{2}}\Big)$$

The plot is attached for your inspection (*solid line theoretical, dots for relative-frequencies*).

We will see several more interesting simulation examples within the **Histogram** and Central Limit Theorem handouts.

PROBLEMS ON SIMULATION OF RANDOM VARIABLES

Sim.1. Explain how you would simulate 3 independent *Weibull* distributed random variables with density $f(w) = 10\, w^4 \exp(-2w^5)\,,\ \ w > 0,$ starting from a sequence of pseudorandom $Unif[0,1]$ variables $U_1, U_2, \ldots$ Your description should be specific enough so that you can give your numerical simulated values if $U_1 = 0.7432$, $U_2 = 0.6545$, $U_3 = 0.2110$.

Sim.2. Explain how you would simulate 5 independent $Binomial(10, 0.25)$ distributed discrete random variable values, starting from a sequence $U_1, U_2, \ldots$ of pseudorandom $Uniform[0,1]$ variables. Again your description should be specific enough to give numerical simulated values if numerical U_k values are given.

Sim.3. Suppose that uniform $[0,1]$ random numbers $U_1, U_2, \ldots, U_{1000}$ are generated on the computer. Let N be the number of the variables U_i, $i = 1, \ldots, 1000$, for which $U_i \geq 0.6$, and let $A = \sum_{i=1}^{1000} e^{-U_i}$.

(a) What is the probability distribution of N ?

(b) To what number would you expect $N/1000$ to be close ?

(c) To what number would you expect $A/1000$ to be close ?

(d) What is the probability distribution or density of *each* of the r.v.'s $\exp(-U_i)$?