Random Number Generation

1. Why all this interest?
2. Methods
3. Pseudo Random Numbers
4. The linear congruential algorithm
5. Non-Uniform Random Nos
6. Analytical Inversion Method
7. Von Neumann's acceptance-rejection methods
8. Testing Random Numbers
9. Problems

Why the Interest In Random Numbers?

As the examples demonstrated we need random numbers for our simulations.

Therefore we need a reliable and efficient source of Random Numbers

Easy? No....

Methods for generating RNs

① Physical Process
- * Radioactive decay
- * Noise in electronic circuits
  - → needs dedicated hardware

② Interface to a Random device
- *for example timing in between Keyboard presses.
  - → slow

③ Pseudo or fake Random Numbers

Pseudo Random Numbers

An algorithm that produces numbers that seem random.

These numbers are not truly Random

If I/P is always the same, then the sequence of numbers produced will be the same.

∴ for a given seed number the same sequence of numbers will always be produced.

By using the system clock we can change the seed.

BUT::
When validating Models in simulations this 'control' on the sequence is USEFULL.

What we know so far

e.g in C

void srand (unsigned seed) → sets the seed value

int rand () → pseudo random number between 0 and RAND_MAX

The sequence is repeatable by calling

srand ()

Changing the Range

There are various ways

eg. new range = [a, a+1, a+2, ... b]

```
(int) (a + (1+b-a) *(double)rand()/ (RAND_MAX+1.0));
```

to generate integers

Avoid the MOD way!

For floating point numbers for [a,b] range

```
a + (b-a) *(double) rand()/RAND_MAX
```

↑ first divide

How can we generate Random numbers?

– The linear congruential algorithm

the sequence of numbers are given by

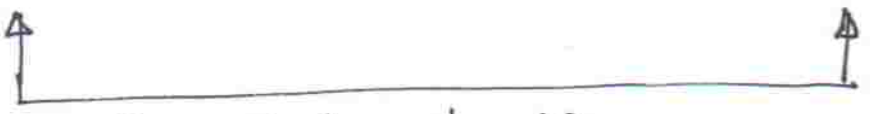
$$I_{j+1} = (a I_j + c) \bmod m$$

Let $a = 7$, $c = 4$, $m = 15$

Seed the sequence with $I_0 = 4$

then $I_{j+1} = (7I_j + 4) \bmod 15$

... 2, 3, 10, 14, 12, 13, 5, 9, 7, 8, 0, 4, 2, 3...



Period length = 12

So the numbers start repeating after 12 draws.

This is close to the maximum theoretical period $m = 15$.

Choose $I_0 = 1$ we have

11, 6, 1, 11, 6, 1

The period is 3 ... useless!

Our generator's performance depends on the seed.

Not a good idea.

Some rules have been developed to guarantee the maximum $(m-1)$ period and all integers in the range $[0, m-1]$ occur before repetition.

① c and m should have no common prime factors.

② $a-1$ is divisible by all the prime factors of m (if there are any)

③ If m is a multiple of 4, $a-1$ should be also.

for example $m = 17$ satisfies the above

Any seed except 5 can be used

$(7 \times 5 + 4) \bmod 17 = 5$ ☹

Unfortunately these criteria are not sufficient to produce a 'good' generator

There may be bad correlation between the numbers, and extensive testing should be carried out on the statistical properties of the sequence produced.

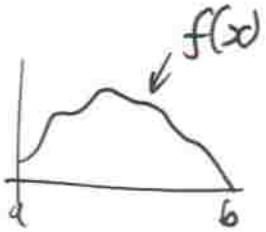
Modern generators use variants of the congruential generator

e.g. combining two sequences or by adding non-linear terms.

Summary

Care needs to be taken by the user when using a random number algorithm in a simulation, for example the period should be much greater than the number of random numbers required in a simulation.

Analytical Inversion Method



① distribution = $f(x)$

properties $f(x) > 0$ for all x

$$\int_{-\infty}^{+\infty} f(x)\,dx = 1$$

② Define the cumulative distribution function

$$F(x) = \int_{-\infty}^{x} f(t)\,dt$$

and its inverse function

$$y = F(x) \iff x = F^{-1}(y)$$

③ u is a random number generated from a uniform distribution $[0, 1]$

We can transform it to a random number in $f(x)$ distribution by calculating

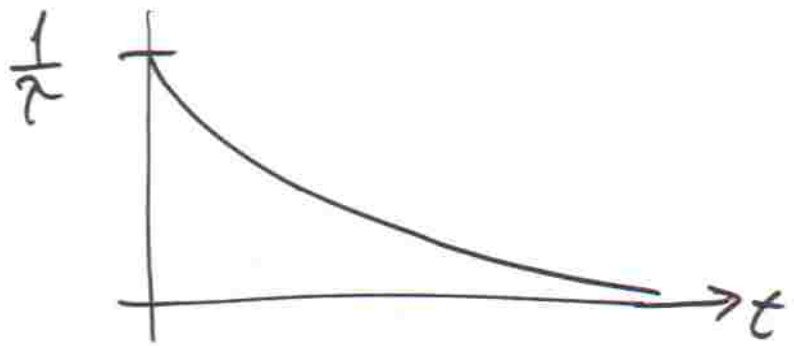
$$x = F^{-1}(u)$$

Example

Required distribution is an exponential decay

$$f(t) = \frac{1}{\tau} e^{-t/\tau} \qquad x > 0$$

$$= 0 \qquad \text{otherwise}$$



$$F(x) = \int_0^x f(t)\,dt = \int_0^x \frac{1}{\tau} e^{-t/\tau}\,dt$$

$$= \left[-e^{-t/\tau}\right]_0^x$$

$$F(x) = 1 - e^{-x/\tau} = y$$

$$\Rightarrow \quad x = F^{-1}(y)$$

$$y - 1 = -e^{-x/\tau}$$

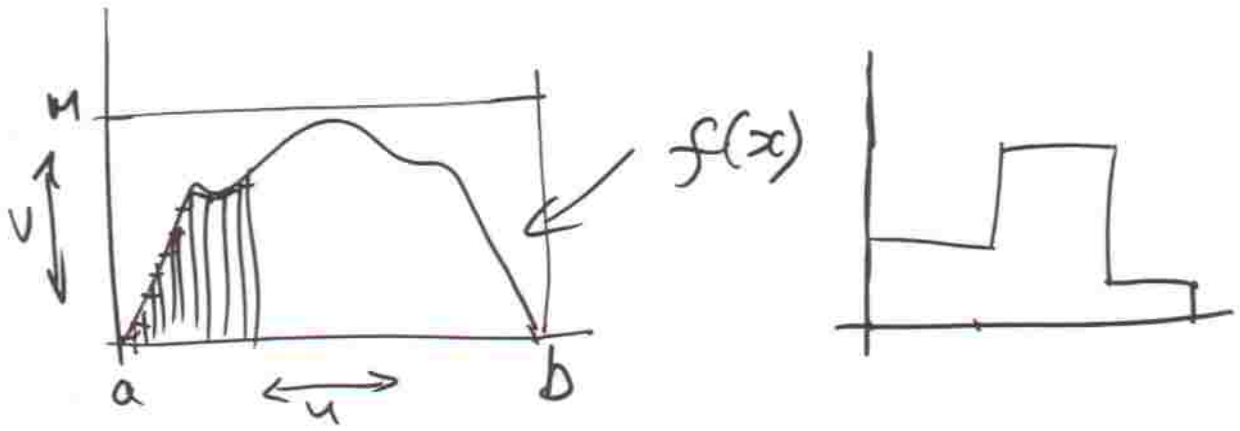
$$1 - y = e^{-x/\tau}$$

$$\ln(1-y) = -\frac{x}{\tau} \ln e^1$$

$$x = -\tau \ln(1-y)$$

$$\left| x = -\tau \ln(y) \right.$$

Von Neumann's acceptance-rejection method



$f(x)$ = required distribution

$h(x) = M \Rightarrow$ constant function

$M > f(x)$ over $[a, b]$

① Generate a random no u from $UD[a, b]$

② Generate a random no v from $UD[0, M)$

③ If $v < f(u)$ accept number u

else reject

④ Go back to step 1