Random Numbers & their Generation

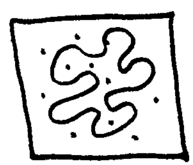
[Tez 95]
[Press, Recipes]

Uses: **Simulation** – randomness in biology
large numbers of individuals
randomness in physics
large numbers of particles
quantum mechanics
randomness in systems
human behavior
large numbers of transactions
...etc.

Testing – too many cases to test exhaustively; e.g. chip testing, hypothesis testing

Monte Carlo evaluation

– to estimate area, volume



prob{inside} ≈ area

A good place to start learning about random number generators, read

"man random"

on your UNIX system. May tell you method, hint at possible problems.

> Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.
> – John von Neumann (1951)

What did von Neumann mean?

We can distinguish between "random"
"pseudorandom"

Pseudorandom numbers have the big advantage of <u>repeatability</u>, for debugging, comparisons, etc.

And they are all-<u>digital</u>, of course.

But they come at a price.

Most popular algorithm is

<u>Linear</u> <u>Congruential</u> <u>Generator</u> (LCG)

$$\boxed{X_n = a\, X_{n-1} + c \quad (\mathrm{mod}\ M)}$$

$$U_n = \frac{X_n}{M}$$ is $\approx$ uniformly distributed in $[0,1)$

or $$U_n = \frac{X_n}{(M-1)}$$ is $\approx$ uniformly distributed in $[0,1]$

LCG's are <u>simple</u>, <u>fast</u>, <u>pretty good most of the time</u>.

<u>Choosing good a, c, M</u>

Notice that X_n are in the range $0 \le X_n < M$.

therefore the sequence $X_1, X_2, \cdots$ must repeat by X_M, and will <u>periodic</u> thereafter.

Conditions are known that ensure this maximum period:

theorem [Tez95] An LCG with parameters a, c, M has period length M if and only if

(i) $\gcd(c, M) = 1$;
(ii) $a = 1 \pmod p$ for every prime p dividing M;
(iii) $a = 1 \pmod 4$ if M is a multiple of 4.

this means we get all integers in $\{0, \cdots, M-1\}$ in some order before repetition, then periodic.

But there are <u>dangers</u> lurking!

... More later

Simple Ways to use RNG's

On my SUN workstation, some nonlinear RNG is used (evidently not LCG).

man entry states range is $0, \ldots, 2^{31}-1 = \text{MAXINT}$

period $\approx 16 * \text{MAXINT}$

Typically, I use:

```
#include <math.h>
#define  MAXINT   2147483647
#define  SEED     122061
long random( );

double  u_random( )
  {  double t;
     t = (double)random( )/ (double) MAXINT ;
     return;
  }

void  init_random( )
  {
     srandom(SEED);
  }
```

or, .../(double)(MAXINT+1) ;
  for [0,1)

→ Some of this may be system dependent!

→ Portability of RNG's is a ~~perennial~~ perennial problem.

- **Choosing an integer i between 1 and N randomly:**

```
i = 1 + (int)( N* u_random( ) );
```

$\underbrace{\texttt{u\_random( )}}$ use [0, 1) version, divide by (MAXINT+1) in u_random

N* u_random( ) is double in range [0, N)

(int) truncates to 0 … N−1

+1 gets to 1 … N

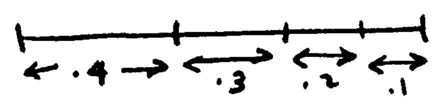
→ If we use [0, 1] version of u_random — one out of 2^{32} cases will yield N+1

this <u>will</u> happen!

If you use arrayofsomething [i] you will dump core!

- **Choosing from a discrete distribution:**

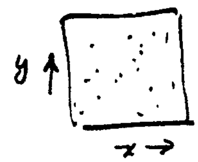
Suppose we want
$p(1) = 0.4$
$p(2) = 0.3$
$p(3) = 0.2$
$p(4) = 0.1$




```
t = u_random( );
if (t < 0.4) {   }
  else if (t < 0.7) {   }
    else if (t < 0.9) {   }
      else {  }
```

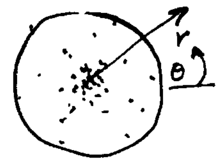
- Pick a random point in 2D:

1. Choose x, y with U-random( )



uniform in unit square

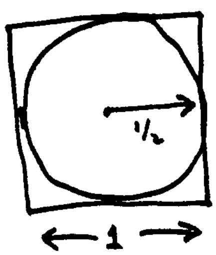
2. choose r uniform 0 to 1, θ uniform 0 to 2π



clustered close to origin

3. How to pick points uniform in "unit" circle? $(r = 1/2)$

rejection method. points discarded if outside



$$\frac{\text{Area unit circle}}{\text{Area unit square}} = \frac{\pi (1/2)^2}{1} = 78.5\%$$

... 3D? etc.

- Shuffle a deck of cards:

to shuffle cards $C_1, \cdots, C_m$:

for $j = m$ down to 2

{ choose random integer from 1 to j, say i

interchange C_i and C_j }

why does this work?

Random number generation is too important
to be left to chance.
– Robert R. Coveyou (1969)

Dangers:

1. Can be disastrous if MAXINT too small

For example, if $MAXINT = 2^{15}-1 = 32{,}767$

with 10^6 calls, sequence is repeated ≈ 30 times!

awful for Monte Carlo

$$\boxed{\text{Moral: use at least 4 bytes, } MAXINT = 2^{31}-1}$$

---

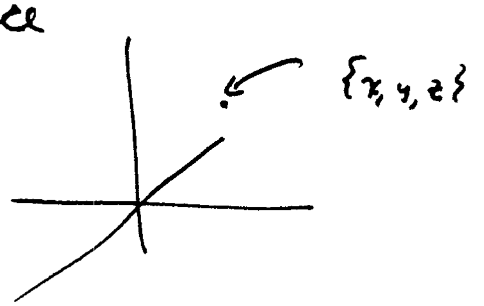
2. Don't use low-order bits!

to get random bit, <u>don't use</u> even/odd test

---

3. Points tend to be serially correlated.

if k successive random numbers are used to plot points in k-space



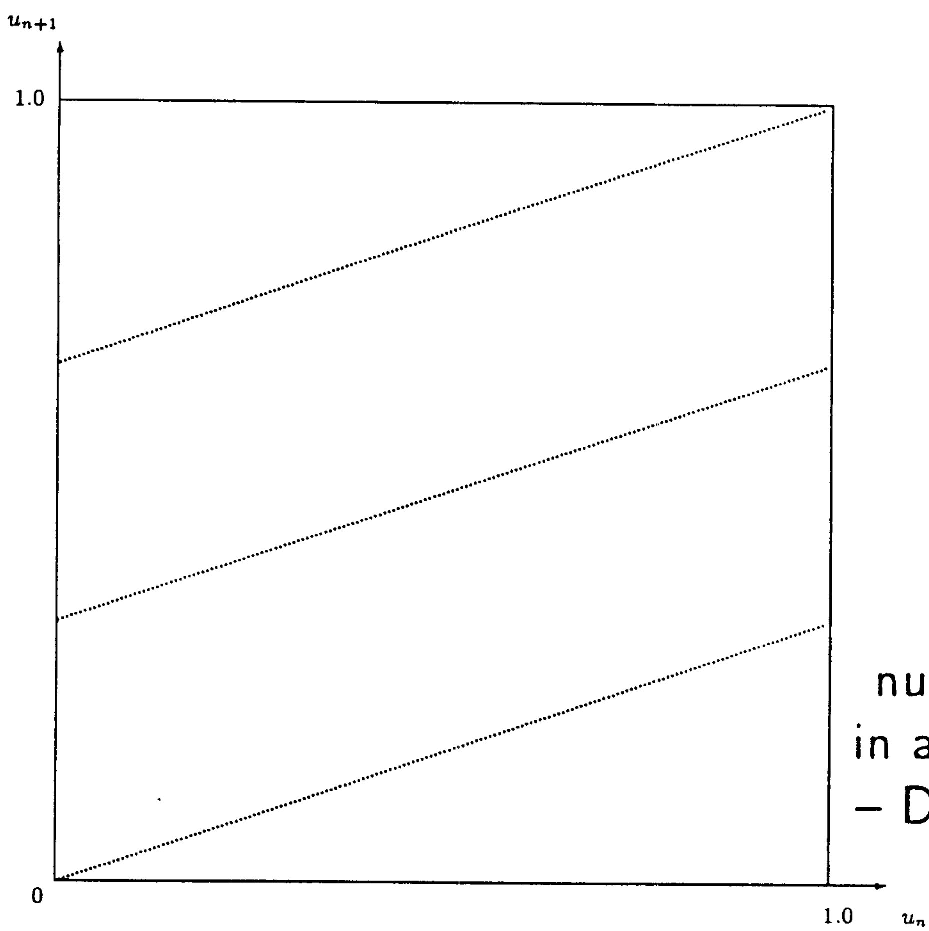
Random numbers fall mainly in the planes.
– George Marsaglia (1968)

points of LCG's tend to lie on <u>at most</u>

$m^{1/k}$ $(k-1)$-D planes.

if $m = 2^{15}$, $m^{1/3} = 32$ UGH!
even $m = 2^{32}$, $m^{1/3} = 1600$ UGH!

old RANDU on IBM mainframes in 60's was badly botched,
Points on only 11 planes!



small example from Tezuda [Tez95]

Figure 3.1 A set of two-dimensional points, $(u_n, u_{n+1}), n = 1, \ldots, 508$, produced by $u_n = X_n/509$, where the LCG is given as $X_n = 170X_{n-1} \pmod{509}$.

...

Every random number generator will fail in at least one application.
– Donald E. Knuth (1969)

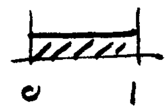
How about other distributions?

... exponential?
... Gaussian?

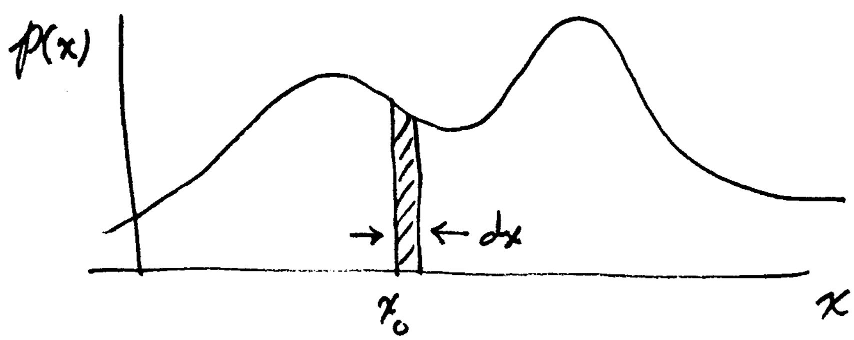
We need some probability theory – quick review

We consider for now <u>continuous</u> random variables

that is, real values, ~~like~~ u_random( )



$p(x)$ = <u>probability</u> <u>density</u> <u>fctn</u> (pdf)

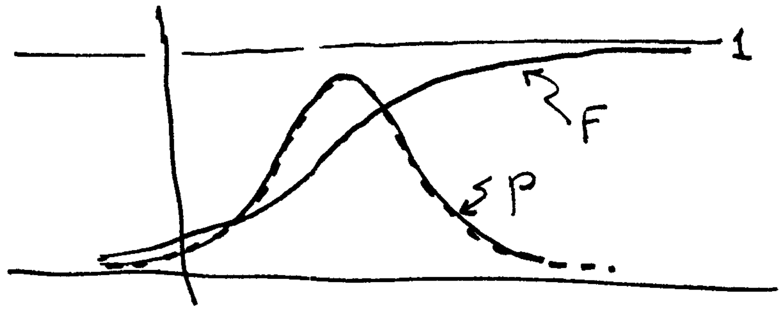


$$\text{prob}(x_0 \le x \le x_0 + dx) = p(x_0) \cdot dx$$

$$\text{always} \begin{cases} p(x) \ge 0 \\ \int_{-\infty}^{\infty} p(x)\,dx = 1 \end{cases}$$

Cumulative Distribution fctn. (cdf)

$$F(y) = \text{prob.}\{x \le y\} = \int_{-\infty}^{y} p(x)dx$$

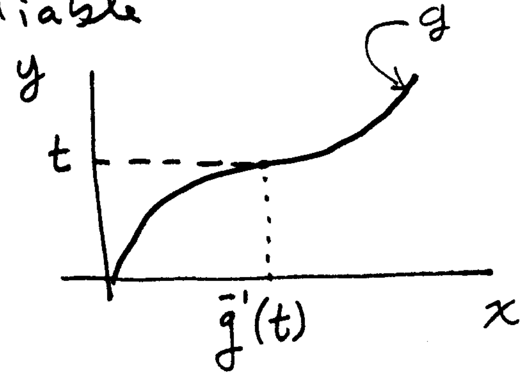


$$p(x) = \frac{dF(x)}{dx}$$

Suppose we generate random x and then compute

$$y = g(x)$$

where $g(\,)$ is monotonic (increasing, say) and differentiable



So $\bar{g}^{1}(\,)$ is well defined, unique.

$$\text{prob}\{y \le t\} = \text{prob}\{x \le \bar{g}^{1}(t)\}$$

Usually, we start with $x = u$ = random( ),
so

$$\text{prob}\{x \le g^{-1}(t)\} = g^{-1}(t)$$

& $\therefore$

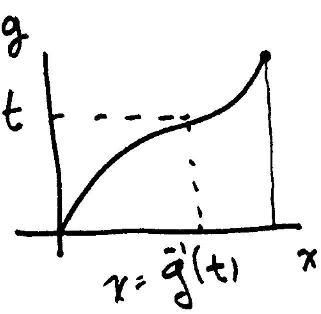
$$\underbrace{\text{prob}\{y \le t\}}_{\text{Cumulative Distr. fctn. for } t} = g^{-1}(t)$$

$$\therefore \quad p(t) = \frac{d}{dt} g^{-1}(t)$$

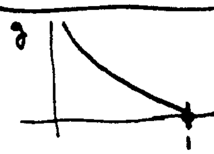
to take care of case when $g()$ is mono. decreasing

$$\boxed{p(t) = \left| \frac{d\, g^{-1}(t)}{dt} \right|}$$

Example 1



Let $g(x) = -\frac{1}{\lambda} \ln x$



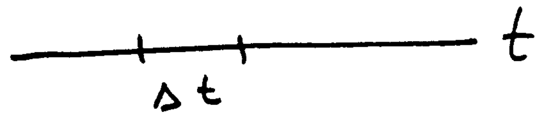
$$x = e^{-\lambda g(x)} = e^{-\lambda t} = g^{-1}(t)$$

$$p(t) = \left| \frac{d\, g^{-1}(t)}{dt} \right| = \left| -\lambda e^{-\lambda t} \right|$$

$$\boxed{p(t) = \lambda e^{-\lambda t}}$$ "exponential" distribution

this is an important distribution.

Physical Interpretation



Suppose an event occurs in a small time interval with prob$\{$event$\} = \lambda \Delta t$

$\lambda =$ expected # events / second

$$\text{prob}\left\{\begin{matrix}\text{exactly } k \text{ events}\\ \text{in time } t\end{matrix}\right\} = \frac{(\lambda t)^k e^{-\lambda t}}{k!}$$

$\longrightarrow$ Poisson distribution

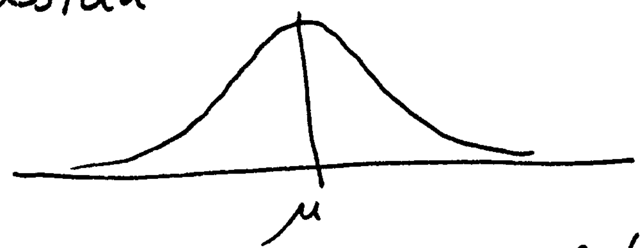
$$\text{prob}\left\{\begin{matrix}0 \text{ events}\\ \text{in time } t\end{matrix}\right\} = e^{-\lambda t}$$

$$\boxed{\begin{matrix}\text{prob. density fctn. of}\\ \text{time } t \text{ to first}\\ \text{event}\end{matrix} = \lambda e^{-\lambda t}}$$

(expected value $= 1/\lambda$)

Example 2 Another very important distribution

Gaussian = Normal = "Bell-shaped"



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

- arises everywhere → sums of small errors that are independent tend to Gaussian
- used to model experimental errors

Generation Method 1 Pick N independent samples using U random(); average

→ quick & dirty, tails no good.

---

transformation method above doesn't work in closed form — in 1-D. (one at a time)

1.2.14

Generation Method 2 — Box-Muller

Generate $x_1 = u\_random()$
$x_2 = u\_random()$

get $y_1 = \sqrt{-2 \cdot \ln x_1} \cos(2\pi x_2)$
$y_2 = \sqrt{-2 \cdot \ln x_1} \sin(2\pi x_2)$

then y_1, y_2 are independent and Gaussian with $\boxed{\mu = 0 \quad \sigma = 1}$

n.b. this works in 2-D & not 1-D because we need to integrate $p(x_1, x_2)$ to get $\bar{g}'$. Can't in 1-D, but can use polar coordinates in 2-D

```
double n_random ()              /* throws second one
{ double u1, u2;                   away */
  double x;
  u1 = u_random ();
  u2 = u_random ();
  x = pow (-2.0 * log(u1), 0.5) *
            cos(2.0* M_PI * u2);

  return (x);

}
```

Note: avoid u_random = 0 this time

1.2.15

the Neave Effect:

TAILS MAY

BAD:

H.R. Neave, "On using the box-muller transformation with multiplicative congruential pseudorandom number generators," Applied Statistics, 22, 92-97, 1973

[Tez95]

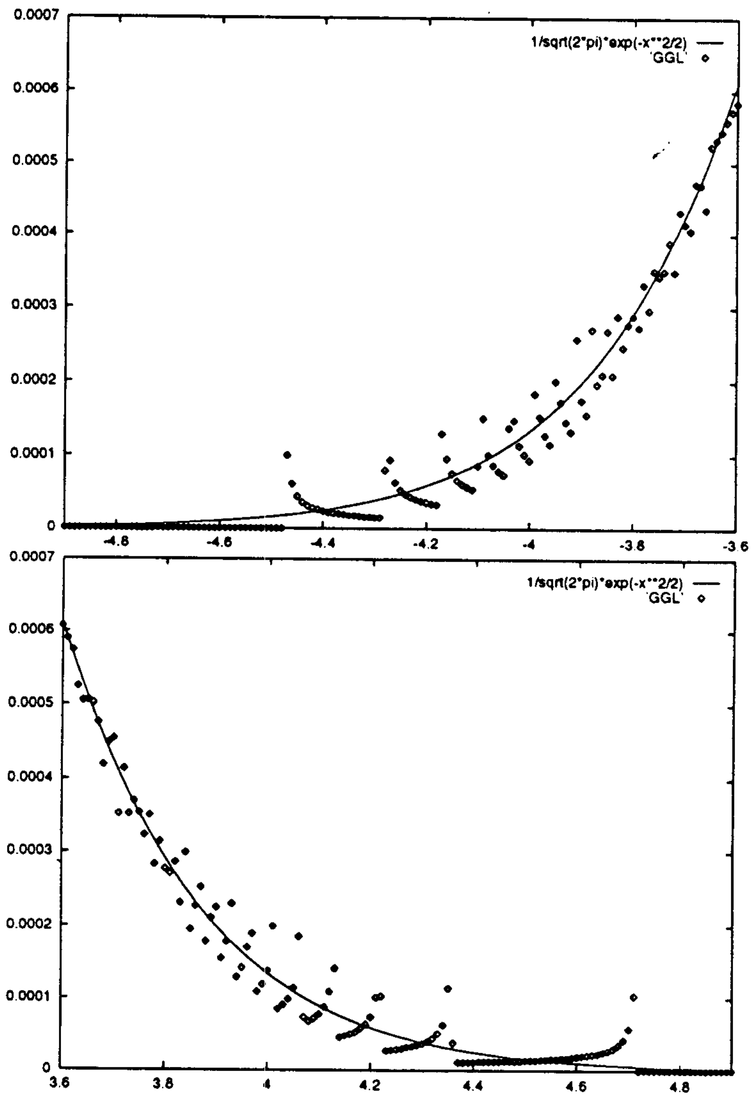


Figure 5.1 Tail distribution in the Box-Muller method with the linear congruential sequence, $X_n = 16807X_{n-1} \pmod{2^{31}-1}$, over the entire period.

Note also truncation $\sqrt{-2\log_e 2^{-32}} \approx 6.66$ upper bound