<u>References</u> Ch. 3 in D.E. Knuth, "The art of computer programming" vol. 2, Addison-Wesley, 1981
W.H. Press, et. al. Numerical Recipes in FORTRAN (also avail. in C), 2^a Ed. Cambridge U. Press, 1992 (available electronically)

"Anyone who considers arithmetic methods of producing random numbers is, of course, in a state of sin"
John von Neumann (1951)

Uses of "Random" numbers

→ Simulation, Sampling, Numerical analysis, Recreation, decision making (e.g. assigning grades - is "3" random?)

Need a <u>sequence</u> of numbers that satisfy certain statistical properties

<u>Mechanical</u>: Dice, roulette etc still in use

<u>Random Physical Processes</u>: e.g. radiactive decay can be used to generate "truly random" numbers

Disadvantages: 1) Impossible to reproduce calculations
2) Speed / convenience

<u>Numerical methods on computers</u>: "pseudorandom" sequences that satisfy most statistical tests of randomness

<u>Generating Uniform Random Numbers</u>

Simple, effective method: Linear congruential

$$X_{n+1} = (\bar{a} \cdot X_n + c) \bmod (m)$$

produces numbers between ϕ and $m-1$

Example: $m = 2^4 = 16$ $a = 13$ $c = 1$

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
X_n	10	3	8	9	6	15	4	5	2	11	0	1	14	7	12	13	10

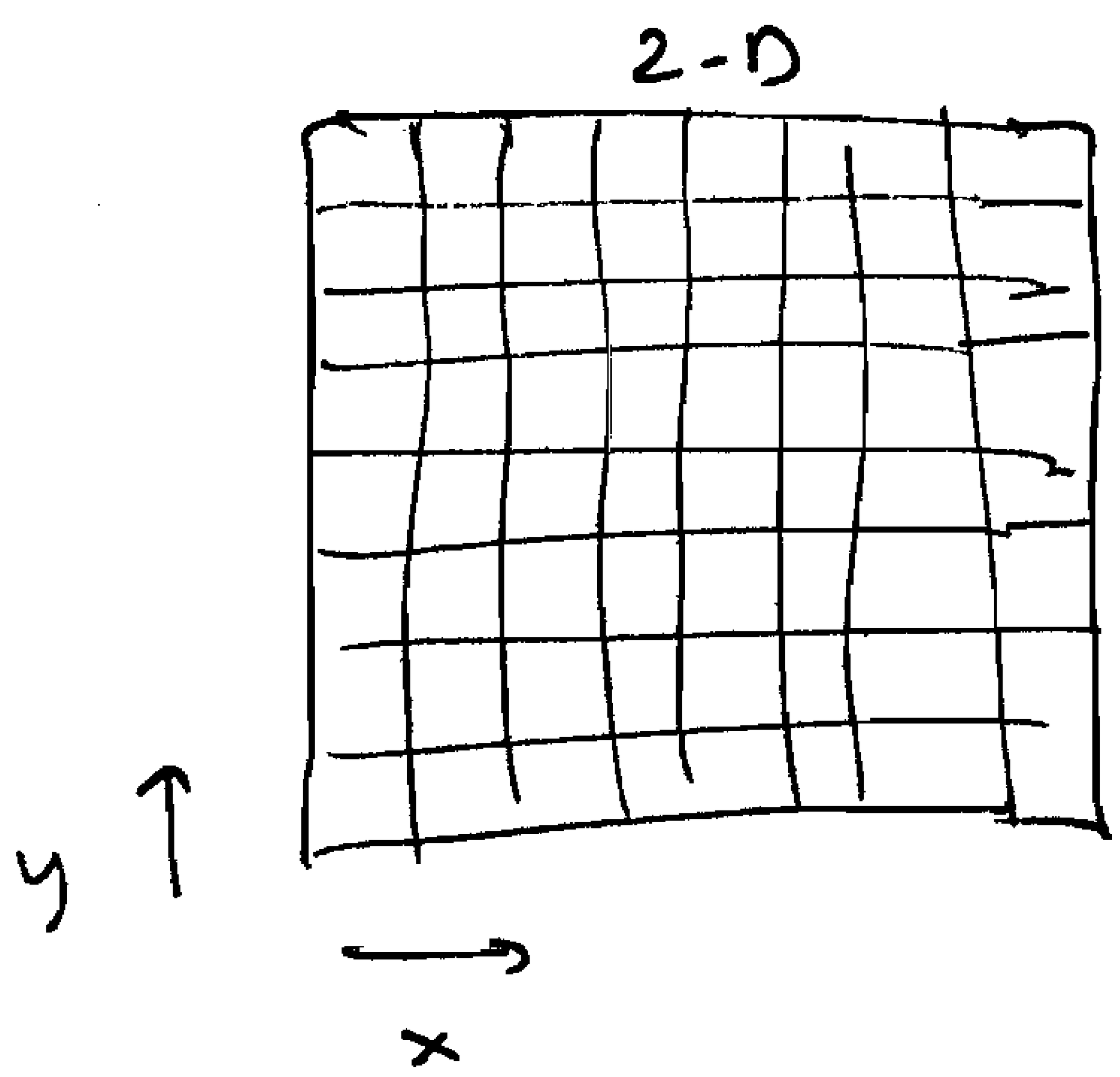
Cycle = 16 always $\leq m$

Practical Example: "Minimal Standard" ran0

$a = 7^5 = 16807$ $m = 2^{31} - 1 = 2147483647$ $c = 0$

period $= 2^{31} - 2 \approx 2.1 \times 10^9$

Spectral test in n dimensions: Generate n random numbers in sequence, place on corresponding "cell" in n-dimensional box:



fraction of empty cells should decrease as e^{-t} t is number of "passes" over the lattice

Simple "minimal standard" generator is said to "fail the spectral test for 2 dimensions for # of samples > $O(10^7)$, much less than the period" $\longrightarrow$ low order serial correlations are present [Num. Recipes Book].

More sophisticated algorithms:

$\rightarrow$ "shuffle" output so that the random number is picked from a short table with e.g. 32 entries. This removes low-order correlations

→ Combine two different sequences so that the new sequence has a period equal to the least common multiple of the two periods:

Add the two sequences, modulo the modulus of either one of them.

$$I_{j+1} = [a_1 I_j \bmod(m_1) + a_2 I_j \bmod(m_2)] \bmod(m_1)$$

E.g. generator ran2 in Numerical recipes has period $\simeq 2.3 \times 10^{18}$, uses shuffling to remove low-order correlations.

From a uniformly distributed sequence, any other "random" function can be recovered.

<u>Application Issues</u>

Consider N_2, suppose we want to model it as a two-center Lennard-Jones system connected by a harmonic spring, with

$$U_b = \frac{1}{2} k \cdot (r - r_0)^2 \qquad k > \emptyset$$

<u>Problem:</u> generate a "random" configuration in a cubic box of length $L \times L \times L$, representative of the state of the system at a certain temperature T.

Placing the first atom is relatively straight-forward:

$x_1 = L * \text{ranf}()$ $y_1 = L * \text{ranf}()$ $z_1 = L * \text{ranf}()$

Placing the second atom is not as easy:

In principle, one could place the second atom at a random position in the box, but the resulting configuration (almost always) will have a very high bond energy U_b. Still, to generate configurations representative of the state at temp. T, we can accept or reject the result with probability

$\exp(-\frac{U_b}{k_B T})$ $\longrightarrow$ generate random number ranf(), compare to $\exp(-\beta U_b)$, if $>$ reject, if $\leq$ accept.

More effective method: generate bond lengths with the correct probability distribution:

Find the maximum and minimum possible bond lengths $r_0+\Delta r$, $r_0-\Delta r$ by setting a "cutoff" probability

$$P \simeq 10^{-10} = \exp\left(-\frac{k(\Delta r)^2}{2k_B T}\right)$$

Generate bond lengths in the interval $r_0-\Delta r, r_0+\Delta r$ uniformly? No!

Probability of a certain bond length must be proportional to r^2! The available volume for placing a second atom is more at longer distances!

→ Generate a number uniformly distributed in the interval $(r_0-\Delta r)^3$ to $(r_0+\Delta r)^3$

→ Take cube root: $p=$ ranf()

$$\delta r = \left[(r_0-\Delta r)^3 * (1-p) + p \cdot (r_0+\Delta r)^3\right]^{1/3} - r_0$$

→ Compute $\exp\left(-\frac{k(\delta r)^2}{2k_BT}\right)$, accept or reject configuration

→ Generate new center position on surface of a sphere of radius $r_0+\delta r$:

[Similar to ranor, algorithm 4g in f+s]

```
ransq = 2.
do while ransq.ge.1
    ran1 = 1-2*ranf()
    ran2= 1-2*ranf()
    ransq= ran1*ran1 + ran2*ran2
enddo
ranh = 2* sqrt(1-ransq)
x2= x1 + ran1*ranh*(r0+δr)
y2 = y1 + ran2 * ranh * (r0+δr)
z2= z1+(r0+δr)* (1-2*ransq)
```

Any alternatives?

Generating the normal (Gaussian) Distribution

For zero mean and unit variance,

$$\rho(x) = \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-x^2}{2}\right)$$

(a) generate uniform random numbers ξ_1, ξ_2 in $(0,1)$

(b) $J_1 = (-2\ln\xi_1)^{1/2}\cos 2\pi\xi_2$, $J_2 = (-2\ln\xi_1)^{1/2}\sin 2\pi\xi_2$

J_1, J_2 are normally distributed

A small variation of this is Algorithm 44 in fts (from Numerical recipes)

```
r=2.
do while (r.ge.2.)
      V1= 2.*ranf() -1.
      V2= 2.*ranf() -1.
      r= V1*V1 + V2*V2
enddo
l= V1*sqrt (-2.*log(r)/r)
```