1. The Monte Carlo method
2. Random number generators
3. The transformation method
4. The acceptance-rejection method
5. Accuracy of MC
6. Uses in particle physics
The Monte Carlo method

What it is: a numerical technique for calculating probabilities and related quantities using sequences of random numbers.

The usual steps:

(1) Generate sequence $r_1, r_2, \ldots, r_m$ uniform in $[0, 1]$.

(2) Use this to produce another sequence $x_1, x_2, \ldots, x_n$ distributed according to some pdf $f(x)$ in which we’re interested. (N.B. $x$ can be a vector.)

(3) Use the $x$ values to estimate some property of $f(x)$, e.g. fraction of $x$ values with $a \leq x \leq b$ gives $\int_a^b f(x) \, dx$.

$\Rightarrow$ MC calculation = integration (at least formally)

Usually trivial for 1-d: $\int_a^b f(x) \, dx$ obtainable by other methods.

MC more powerful for multidimensional integrals.

MC $x$ values = ‘simulated data’

$\rightarrow$ use for testing e.g. statistical procedures.
Random number generators

Goal: uniformly distributed values in $[0, 1]$.  
Toss coin for e.g. 32 bit number … (too tiring).
⇒ ‘random number generator’
= computer algorithm to generate $r_1, r_2, \ldots , r_n$.

Example: the multiplicative linear congruential generator (MLCG)

$$n_{i+1} = (an_i) \mod m , \quad \text{where}$$

\[ n_i = \text{integer} \]
\[ a = \text{multiplier} \]
\[ m = \text{modulus} \]
\[ n_0 = \text{seed} \]

N.B. $\mod = \text{modulus (remainder), e.g. 27 mod 5 = 2.}$

The $n_i$ follow periodic sequence in $[1, m - 1]$.

Example (cf. Brandt): $a = 3, m = 7, n_0 = 1$:

\[
\begin{align*}
n_1 &= (3 \cdot 1) \mod 7 = 3 \\
n_2 &= (3 \cdot 3) \mod 7 = 2 \\
n_3 &= (3 \cdot 2) \mod 7 = 6 \\
n_4 &= (3 \cdot 6) \mod 7 = 4 \\
n_5 &= (3 \cdot 4) \mod 7 = 5 \\
n_6 &= (3 \cdot 5) \mod 7 = 1 \leftarrow \text{sequence repeats!}
\end{align*}
\]

Choose $a, m$, to obtain long period (maximum $= m - 1$).
Random number generators (continued)

\[ r_i = \frac{n_i}{m} \] are in \([0, 1]\) (0 and 1 excluded), but are they ‘random’???

Choose \(a\), \(m\), so that the \(r_i\) pass various tests of randomness:

- Uniform distribution in \([0, 1]\)
- All pairs independent (no correlations)

e.g. L’Ecuyer, Commun. ACM 31 (1988) 742 suggests

\[ a = 40692 \]
\[ m = 2147483399 \]

Test with 10000 generated values:

Far better algorithms available e.g. **RANMAR**, period \(\approx 2 \times 10^{43}\).

For more info see e.g.

F. James, Comput. Phys. Commun. 60 (1990) 111;
Brandt, chapter 4.
program TEST_RANMAR

implicit NONE

c Needed for HBOOK routines

integer hsize
parameter (hsize = 100000)

integer hmemor (hsize)
common /pawc/ hmemor

c Local variables

character*80 outfile
integer i, icycle, istat
integer NTOTIN, NTO2IN, IJKLIN
real rvec(1) ! vector of random nos. (here only 1)

c Initialize HBOOK, open histogram file, book histograms, set seed.

call HLIMIT (hsize)
outfile = 'test_ranmar.his'
call HROPNEN (20, 'histog', outfile, 'N', 1024, istat)
call HBOOK1 (1, 'uniform dist', 100, 0., 1., 0.)

call RMARIN(IJKLIN,NTOTIN,NTO2IN) ! sets initial seed

write (*, *) 'Enter initial seed between 0 and 900 000 000'
read (*, *) IJKLIN
NTOTIN = 0
NTO2IN = 0
call RMAEIN(IJKLIN,NTOTIN,NTO2IN)

c Generate 10000 values, enter into histogram, then store histogram.

do i = 1, 10000
   call RANMAR (rvec, 1)
   call HF1 (1, rvec(1), 1.)
end do

call HR10UT (0, icycle, ' ')
call HRE0ND ('histog')

stop
END
The transformation method

Given \( r_1, r_2, \ldots, r_n \) uniform in \([0, 1]\), find \( x_1, x_2, \ldots, x_n \) which follow \( f(x) \) by finding a suitable transformation \( x(r) \).

\[
\int_{-\infty}^{r'} g(r) \, dr = r' = \int_{-\infty}^{x(r')} f(x') \, dx' = F(x(r'))
\]

That is,

\[
\text{set } F(x(r)) = r \text{ and solve for } x(r).
\]
Example of the transformation method

Exponential pdf: \( f(x; \xi) = \frac{1}{\xi} e^{-x/\xi} \quad (x \geq 0) \)

Set \( \int_0^x \frac{1}{\xi} e^{-x'/\xi} \, dx' = r \) and solve for \( x(r) \).

\[ x(r) = -\xi \log(1 - r) \quad \left( x(r) = -\xi \log r \text{ works too.} \right) \]
The acceptance-rejection method (von Neumann)

Enclose the pdf in a box:

(1) Generate a random number \( x \), uniform in \([x_{\min}, x_{\max}]\), i.e. \( x = x_{\min} + r_1(x_{\max} - x_{\min}) \) where \( r_1 \) is uniform in \([0, 1]\).

(2) Generate a second independent random number \( u \) uniformly distributed between 0 and \( f_{\max} \), i.e. \( u = r_2 f_{\max} \).

(3) If \( u < f(x) \), then accept \( x \). If not, reject \( x \) and repeat.

Example:

\[
f(x) = \frac{3}{8} (1 + x^2)
\]

\((-1 \leq x \leq 1)\)
Accuracy of Monte Carlo

MC calculation = integration.
Compare to trapezoidal rule,
\( n = \) number of computing steps.

For 1-dimensional integral:

**MC:** \( n \propto \) number of random values generated
accuracy \( \propto 1/\sqrt{n} \)

**Trapezoid:** \( n \propto \) number of subdivisions
accuracy \( \propto 1/n^2 \)

Trapezoid wins! But in \( d \) dimensions this becomes

**MC:** accuracy \( \propto 1/\sqrt{n} \) ← independent of \( d \)!

**Trapezoid:** accuracy \( \propto 1/n^{2/d} \)

MC wins for \( d > 4 \). Gaussian quadrature better than trapezoid,
but for high enough \( d \), MC always wins.
Monte Carlo event generators

Simple example:

\[ e^+ e^- \rightarrow \mu^+ \mu^- \]

Generate \( \theta \) and \( \phi \):

\[
f(\cos \theta; A_{FB}) \propto (1 + \frac{8}{3} A_{FB} \cos \theta + \cos^2 \theta)
\]

\[
g(\phi) = \frac{1}{2\pi}
\]

Less simple examples:

\[ e^+ e^- \rightarrow \text{hadrons: } \]

JETSET (PYTHIA)  
HERWIG  
ARIADNE  

\[ pp \rightarrow \text{hadrons: } \]

ISAJET  
PYTHIA  
HERWIG  

\[ e^+ e^- \rightarrow \text{WW: } \]

KORALW  
EXCALIBUR  
ERATO  

Output = ‘events’, i.e. for each event, a list of final state particles and their momentum vectors.
Monte Carlo detector simulation

Takes as input the particle list and momenta from generator.

Simulate detector response:

multiple Coulomb scattering (generate scattering angle)
particle decays (generate lifetime)
ionization energy loss (generate $\Delta$)
EM/hadronic showers
production of signals, electronics response

Output = simulated raw data
$\rightarrow$ input to reconstruction software (track finding/fitting, etc.)

Uses:

Predict what you should see at ‘detector level’ given a certain hypothesis for ‘generator level’. Compare with the real data.

Estimate various ‘efficiencies’ $= \frac{\# \text{ events found}}{\# \text{ events generated}}$

Programming package: GEANT
1. The Monte Carlo method

   numerical technique for computing probabilities (and things that can be related to probabilities) using random numbers,
   MC ↔ integration,
   does not depend on interpretation of probability.

2. Random number generators

   produce sequence \( r_1, r_2, \ldots, r_n \) uniform in \([0, 1]\),
   actually pseudorandom (i.e. reproducible if use same seed),
   simple algorithm: MLCG,
   better ones exist, e.g. RANMAR.

3. The transformation method

   set cumulative distribution \( F(x) = r \), solve for \( x(r) \),
   produces one value of \( x \) for each value of \( r \).

4. The acceptance-rejection method

   must be able to enclose pdf \( f(x) \) in a box,
   algorithm slow if pdf is sharply peaked.

5. Accuracy of MC

   accuracy \( \propto 1/\sqrt{n} \)

6. Uses in particle physics

   event generators
   detector simulation