Monte Carlo methods

First: hit-or-miss Monte Carlo, determine $\pi$

- Throw random darts on a square board with a circle in it
- Count #hits in the square and in the circle
- $\# \text{hits in circle} / \# \text{hits in square} = \pi / 4$

Computer approach:

```c
for (i=0; i<N; i++) {
    x=random();
    y=random();
    if (x^2+y^2<1) hits_c++;
}
printf("Pi=%lf\n", hits_c*4.0/N);
```
Second: Monte Carlo integration to determine $\pi$

Function $f(x) = \sqrt{1-x^2}$, $-1 \leq x \leq 1$

Integral of $f(x)$ equals $\pi/2$

- Take random numbers $r_1 \ldots r_N$
- Average of $r_i$ equals $\pi/4$

Computer approach:

```c
for (i=0; i<N; i++) {
    x=random();
    sum+=sqrt(1-x^2);
}
printf("Pi=%.1f\n", sum*4.0/N);
```
Both of these require uniform random numbers.

How do we generate pseudo-random numbers?

a common algorithm: linear congruential RNG

\[ i_n = (a \cdot i_{n-1} + c) \mod m \]

One possible choice:

\[ a = 2416, \ c = 374441 \text{ and } m = 1771875 \]

If you do multiple runs and want improved statistics, use different seeds (here \( i_0 \) is the seed). One possibility:

\[ i_0 = \text{number of seconds since Jan 1}^{\text{st}}, 1970 \]

(subroutine time() in <times.h>)

Provided you do not restart within a second.

Note: do not reseed the RNG during a run!
Sequence repeats itself after m calls (or fewer)

How random are RNGs?

Charmaine Kenny (2005) recommended the following tests from the NIST suite for use on RANDOM.ORG:

- Frequency Test: Monobit
- Frequency Test: Block
- Runs Test
- Test for the Longest Runs of Ones in a Block
- Binary Matrix Rank Test
- Discrete Fourier Transform (Spectral Test)
- Non-Overlapping Template Matching Test
- Overlapping Template Matching Test
- Maurer's Universal Statistical Test
- Linear Complexity Test
- Serial Test
- Approximate Entropy Test
- Cumulative Sums Test
- Random Excursions Test
- Random Excursions Variant Test
Visualisation often helps as well, for instance

[Bo Allen, with rand() from PHP in Microsoft Windows]
- Your task: perform tests on your RNG, and present results with proper statistical analysis in the SOM.
- Lectures of Deb Panja will prepare you for this.
- Your grade depends on this!

Possible projects:

Compare performance of MC integration with other integration methods, on a variety of d-dim. integrals:

- Speed of convergence in d dimensions
- Ways to boost MC integration, for instance automated “importance sampling”? (key words VEGAS, MISION, recursive stratified sampling, .....)
Importance sampling

Some mathematics:

\[ \frac{\partial G(x)}{\partial x} = g(x) \]
\[ \implies \int f(x) \, dx = \int \frac{f(x)}{g(x)} \, dG(x) \]

In practice, this means that if there is an invertible function \( g(x) \) such that \( f(x) \sim g(x) \), one can compute the integral over \( f(x) \) more accurately using:

- Sample points \( x \) from (normalized) distribution \( g(x) \)
- Determine the average \( <f(x)/g(x)> \)

Note 1: if \( g(x) \) is almost identical to \( f(x) \), then the spread in \( f(x)/g(x) \) is small, hence the statistical error.

Note 2: this allows integration over infinite domains

Warning: integration of functions with a singularity can give complications