MATHEMATICAL ALGORITHMS I

1. Simple algorithms for pseudo-random generators

In this problem section you will need to write a lot of small functions. It is probably best to move them into smaller separate files and use header files to include them into your main program.

1.1. **Problem: Linear congruence generators.**

(1) Implement the linear congruence generators. Try to find numbers of the form a, c and $m = 2^w$ that work well.
(2) Make this algorithm generate a $[0, 1]$-uniformly distributed pseudo-distribution.
(3) Test the mean and standard deviation

1.2. **Problem: Basic tests.** Make some testing tools for your pseudo-random generators: include covariance between $cov(X_i, X_{i+1})$, and the Kolmogorov-Smirnov test. It may be useful to practice with function pointers to easily switch between different generators.

Run these tests using the a uniform distribution outputted by the linear congruence generators from the previous problem.

1.3. **Problem: Middle square method.** Design and implement an algorithm for generating a sequence with the middle square method. Use a `long int` as an integer.

(1) make this algorithm generate a $[0, 1]$-uniformly distributed pseudo-distribution.
(2) test the mean and standard deviation
(3) Run your tests. How does it do for different seeds?

1.4. **Problem: Lagged Fibonacci.** Implement the lagged Fibonacci generator (allowing k and ℓ to vary).

(1) Consider the case $\ell = 24$, $k = 55$. Run the tests from your earlier problem. How does it do?

1.5. **Problem: Other distributions.** Suppose that X is a uniformly distributed random variable with values in $\{0, \ldots, n-1\}$, so

$$P(X = x) = \begin{cases} \frac{1}{n} & x \in \{0, \ldots, n-1\} \\ 0 & \text{otherwise.} \end{cases}$$

(1) What is the distribution of $Y = X \mod k$? In other words, give $P(Y = y)$.
(2) Implement an algorithm that returns a uniformly distributed pseudo-random variable Y with values in $\{0, \ldots, k-1\}$.

1.6. **Problem: Normal distribution.** The probability density of the normal distribution with mean $\mu = 0$ and standard deviation $\sigma = 1$ is given by

$$p(t) = \frac{e^{-t^2/2}}{\sqrt{2\pi}}$$

(1) Write a program returning a normally distributed pseudo-random variable using the algorithm described in class: for the uniformly distributed pseudo-random variable use a linear congruence sequence.

(2) Use the Taylor theorem to approximate the distribution function

$$F(x) = P(X \leq x) = \int_{-\infty}^{x} \frac{e^{-t^2/2}}{\sqrt{2\pi}} dt.$$

by a polynomial of degree n. Write a function that computes the value of this polynomial and plot the graph of this polynomial.

(3) Use the Kolmogorov-Smirnov test to see whether this is indeed useable as a pseudo-random generator for the normal distribution. How does it vary with the linear congruence generator?

Download a basic svg-library from the eTl-page to plot the curve.