Pseudo Random Number Generator

CS4050 – Assignment 4

Ryan Flageolle

**Description**

There are many different algorithms used to generate pseudo-random numbers, I looked into the BlumBlumShub algorithm and compared it to the Mersenne twister algorithm which is used as the built-in random library for python. To test the “randomness” of each I compared mean, standard deviation and variance of each, and to test the sequences I tested each generator using the frequency and runs test as described in the NiST test suite.

**BlumBlumShub**

The BlumBlumShub is a one-way function derived from Michael O. Rabin.

xn+1 = x2n % M

In my implementation, I set a seed (x0) to the value of microseconds from the current time, then for some number n I calculate the next value of x and mod the answer by 2 to return a binary string. From this binary string I take a substring and return its integer representation. This number generator does not produce cryptographically safe random numbers however it does seem to produce similarly random numbers as python’s Mersenne twister algorithm.

**Tests Used**

I decided to test my sequences using the frequency test and runs test as they are explained in the NiST test suite. I chose these tests because they determine the “randomness” of a prng (pseudo-random number generator) by assessing the sequence of bits generated. The frequency test determines whether the sequence given has a frequency of 1s to 0s which is close to the expected rate of .5. The runs test determines whether a given sequence of bits is random by counting the number of runs, changes between 1s and 0s for example 0011011001 has 6 runs [00, 11, 0, 11, 00, 1], and comparing it to the number of expected runs.

**Results**

mean of my algorithm after 500,000 runs is: 511.623306

mean of built in python algorithm after 500,000 runs is: 512.034056

Standard Deviation of my algorithm after 500,000 runs is: 313.22502074966866

Standard Deviation of built in python after 500000 runs is: 295.5963197338371

My algorithm passes the frequency test: 100%

Built in python algorithm passes the frequency test: 97%

My algorithm passes the runs test: 98%

Built in python algorithm passes the runs test: 94%

**Conclusions**

These results lead me to believe that my interpretation of the Blum Blum Shub algorithm