Basics of Monte Carlo Simulations 2021
Report Ex2

Jake Muff 8/2/2021

Question 1

Writing a quadaratic congrutential generator. The solution code for this is found in QCG.cpp, with the function found in the header file rng.h under rand_qcg(). For this I picked

$$m = 2401, a = 7, b = 1, c = 8$$

m was picked for this as it is 7^4. These numbers were not chosen in order to make the period long but to achieve and find necessary values in order for the QCG to work. The interval was calculate and found to match what was expected. This is found in rand_qcg_interval in rng.h.

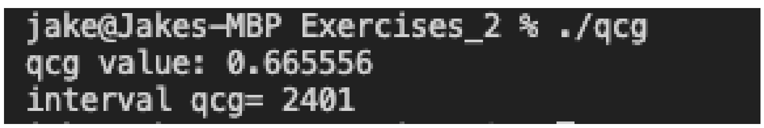


Figure 1: Screenshot of output of QCG.cpp showing that interval is equal to m

Question 2

Understanding the TGFSR.
Start off with:

$$k = 0 \rightarrow a_k = 10$$

Bitshift a_k by converting to binary and shifting 1 along

$$a_k = 10_{10} = 1010_2 \rightarrow 101_2 = 5_{10}$$

Computing $(k + q)\ mod\ p$

$$(k + q)\ mod\ p = (0 + 11)\%25 = 11$$

The index of $(k + q)\ mod\ p$ is then

$$a_{(k+q)\ mod\ p} = 21$$

Computing the first XOR :

$$21\ XOR\ 5 = 16$$

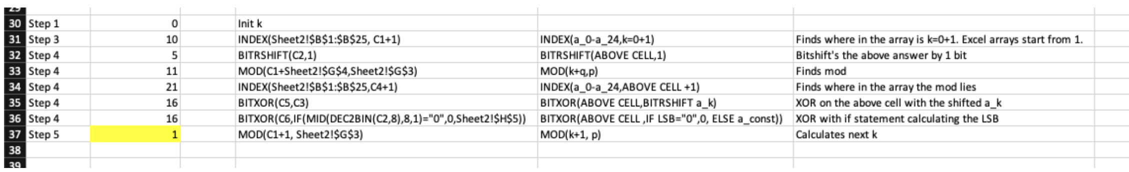
The least significant bit of a_k is 0 so the second XOR is:

$$16\ XOR\ 0 = 16$$

$$k = k + 1\%25 = 1$$

$$a_{k'} = 16$$

Thus the new value assigned to a_0 is 16.



30	Step 1	0	Init k		
31	Step 3	10	INDEX(Sheet2!$B$1:$B$25, C1+1)	INDEX(a_0-a_24,k=0+1)	Finds where in the array is k=0+1. Excel arrays start from 1.
32	Step 4	5	BITRSHIFT(C2,1)	BITRSHIFT(ABOVE CELL,1)	Bitshift's the above answer by 1 bit
33	Step 4	11	MOD(C1+Sheet2!$G$4,Sheet2!$G$3)	MOD(k+q,p)	Finds mod
34	Step 4	21	INDEX(Sheet2!$B$1:$B$25,C4+1)	INDEX(a_0-a_24,ABOVE CELL +1)	Finds where in the array the mod lies
35	Step 4	16	BITXOR(C5,C3)	BITXOR(ABOVE CELL,BITRSHIFT a_k)	XOR on the above cell with the shifted a_k
36	Step 4	16	BITXOR(C6,IF(MID(DEC2BIN(C2,8),8,1)="0",0,Sheet2!$H$5))	BITXOR(ABOVE CELL ,IF LSB="0",0, ELSE a_const))	XOR with if statement calculating the LSB
37	Step 5	1	MOD(C1+1, Sheet2!$G$3)	MOD(k+1, p)	Calculates next k
38					

Figure 2: I calculated this in excel. Here is a screenshot

Question 3

Under the Chi2.cpp program you will find a program to output the random numbers to .txt using the various array functions found in rng.h. It was easier to take the arrays of random numbers and computer the χ^2 value in python.

As we can see by the uniform distirbutions. The MT and PM are more 'noisy' and less uniform whereas the LCG and QCG are fairly uniform. This is shown by the chi2 values calcuated. The MT and PM generate good χ^2 values whereas the LCG and QCG do not. Unfortunately this was done for just 1 run and thus the average is not very accurate.

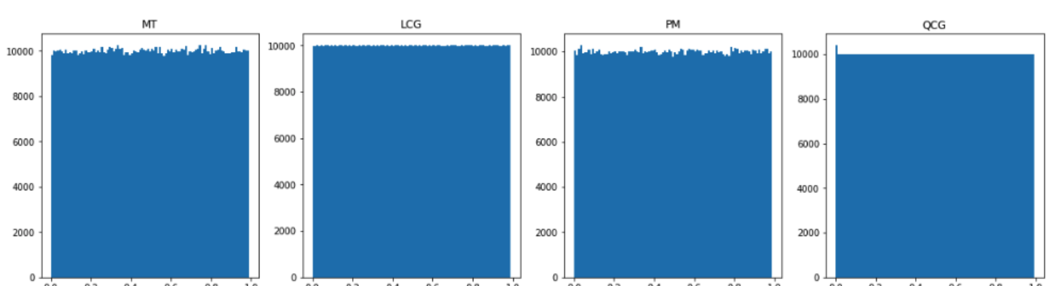


Figure 3: Uniform Distribution of the $1e6$ random numbers

```
MT chi square: 107.808051
LCG chi square: 0.746378
PM chi square: 85.009927
QCG chi square: 16.466196
```

Figure 4: χ^2 sum for each of the Distribution

```
(base) jake@Jakes-MBP Exercises_2 % python chi2.py
Expected median: 98.334828659599
mean chisq: 108.69180000000001
median chisq: 108.6918
chi ^2 for MT : 107.80805051541493
chi ^2 for LCG : 0.7463776136121061
chi ^2 for PM : 85.0099268167095
chi ^2 for QCG : 16.46619551966649
(base) jake@Jakes-MBP Exercises_2 %
```

Figure 5: χ^2 sum for each of the Distribution

Question 4

Performing the autocorrelation test for LCG and Park-Miller Generators. This is shown in autocorr_test.cpp. This file runs by specifying your output file and the number of points (10^5) on the run line from the command line. The program works by splitting the autocorrelation test formula into the mean, mean squared and the variance. The program then outputs the autocorrelation results along with the index number to a .dat file. The .dat file is then read in and plotted with pyplot as shown in the plot_q3.py

As you can see with the plots the LCG is not as good as it's PM counterpart. Note that I also included the inbuilt rand() function as well as the Mersenne Twister for comparison. Interestingly the LCG shuffle did not change the results for the autocorrelation. The result for the original LCG is to be expected because the values will correlate after its period.

To calculate the LCG shuffle (implemented in rng.h) I used the Bays-Durham shuffled used in ran1 in Numerical Recipes but adapters for LCG.

```
4 warnings generated.
jake@Jakes-MBP Exercises_2 % ./autocorr test.dat 100000
You have entered 3 arguments:
./autocorr
test.dat
100000
```

Figure 6: Command Line arguments shown for autocorrelation

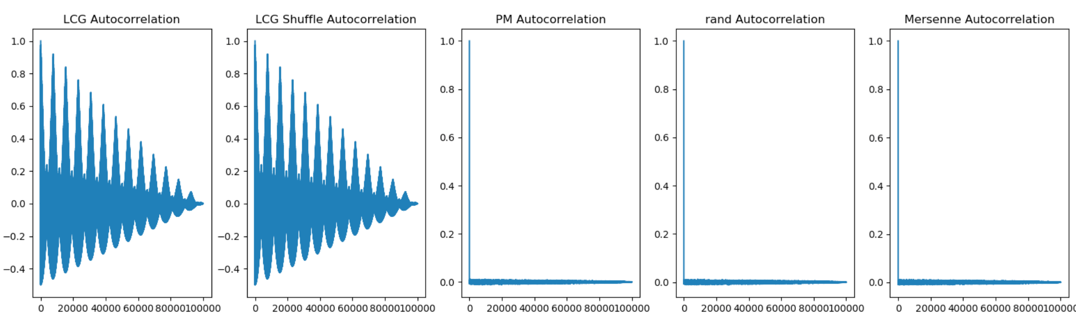


Figure 7: Plot for the results of the autocorrelation