Tutorial 1 – Giovanni Filomeno - 12315325

# Exercise 1

## Compute by hand the resulting sequence of pseudo-random numbers

Since the part (b) asks for the period of the sequence, I will evaluate by hand sequence of pseudo-random numbers until I have again the same number. The general formula is with .

## What is the period of the sequence?

Since I obtain again 19 after 15 iterations, the period is 15.

## Why does it not have maximum period ?

Because the period is the theoretical maximum period given , however not every multiplier is a primitive root of mod 31. Finding a period of 15 leads to the following congruence which is true if I compute it by hand .

## Find a value for such that the generator has maximum period

I can use the formula where and I try some proper divisor of 30 for example . Doing this, I found that , which leads to the following formula .

# Exercise 2

## Implement the linear congruential generator

The screenshot of the R-code is provided in Figure 1.

|  |
| --- |
|  |
| Figure 1: Linear Congruential Generator |

## Run the function using values from Exercise 1

The code in exercise 2.a gives the following sequence:

|  |
| --- |
|  |
| Figure 2: Sequence |

# Exercise 3

## What is the period of each generated sequence?

I used the same code of Exercise 2 three times, using the command *length().* The results are: 8190 (A), 4095 (B), 4095 (C).

## Plot all pairs of consecutive numbers and consider the auto-correlation for each of the three cases.

The Figure 3 shows the plot of all pairs of consecutive numbers for a=17, a=29, and a=197.

|  |
| --- |
|  |
| (a) |
|  |
| (b) |
|  |
| (c) |
| Figure : Plots of consecutive numbers |

Additionally, as requested from the exercises, I also plot the auto-correlation, however, since the correlation in each sequence is pretty small, the three graphs look the same even though their periods differ (8190 vs 4095). Therefore, they do not add any meaningful information. However, just for sake of completeness, I also added in Figure 4 the plot coming from the function *acf()* for a=17.

|  |
| --- |
|  |
| Figure : ACF plot for a=17 |

## What is the best choice of multiplier a between (A), (B) and (C)?

I would take the multiplier of (A), a=17, since the period is larger, so it means a repetition comes later, even if visually it tends to form straight lines.

# Exercise 4

## Write function *Fks(x)*

|  |
| --- |
|  |
| Figure : Fks(x) function |

## Define function *Kalpha(alpha)*

|  |
| --- |
|  |
| Figure : Kalpha(alpha) function |

## Write function *Dno(A)*

|  |
| --- |
|  |
| Figure : Dno(A) function |

## Write function *pval(A)*

|  |
| --- |
|  |
| Figure : Dno(A) function |

## Write function *ourkstest(A,alpha)*

|  |
| --- |
|  |
| Figure : ourkstest(A,alpha) function |

## Run the code

|  |
| --- |
|  |
| Figure : Run the two examples |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Figure : Output of the previous code | | |

My functions give in both tests the same final decision w.r.t. to buil-in function. At I do not reject H0 ( for my functions, for the build-in. Both ). At I reject H0 ( for my functions, for the build-in. Both ). The numerical difference comes from the truncation () I have in the function.

# Exercise 5

|  |
| --- |
|  |
| Figure : Quadratic Congruential Generator function |

# Exercise 6

## Find *d* and *a*

To find the optimal parameters, I simply hard-coded a double for-loop (see Figure 13), which gives me and .

|  |
| --- |
|  |
| Figure : Double for to find the parameters |

## Perform a test

After finding the requested *a* and *d*, I generated the sequence, and I normalized the sequence over *m* and then I applied the build-in Kolmogorov Smirnov test which gives me and .

## Plot 1000 pairs

|  |
| --- |
|  |
| Figure : 1000 pairs of corresponding consecutive numbers |