Stochastic Simulation

Georgios Tsimplis (s200166)

June 2020

Contents

1 Exercise 1. Generation of Random Numbers

1.1 Pseudo-random number Generator

In this section we will implement a pseudo-number generator and after this we will test it with chi-squared test, Kolmogorov-Smirnov test as well as Run tests I, II, III and correlation test. This sequence of different values are deterministically created using specific techniques in order to give us the sense of being independent and uniform random numbers from the interval [0,1]. The formula of the Linear Congruential Generator that was used is the below:

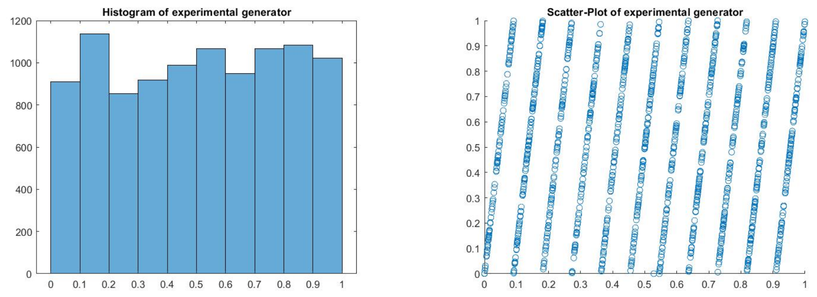
$$x_n = mod(ax_{i-1} + c,\, M)$$

$$U_i = \frac{x_i}{M}$$

In order to have a maximum cycle length it is necessary to implement the following theorem.
Theorem: The Linear Congruential Generator has a full length if and only if:

1. M and c are relative prime
2. For each prime factor p of M we have $mod(a, p) = 1$
3. if 4 is a factor of M then $mod(a, 4) = 1$ and if M is prime then we obtain a full period only if $a = 1$

In the implementation we set $a = 11$, $c = 1$, $M = 10001$ and we started from $x_0 = 1$. We generated 10000 pseudo-random numbers which are presented in the following plots.



(a) Histogramme

(b) Scatter Plot

Figure 1

1.2 Statistical Tests

One Statistical tool which measures the difference between the observed and the expected values is the χ^2**-test** and formulated by Karl Pearson. For this test can be proven that under some specific circumstances is has asymptotically χ^2-distribution with $n-1-m$ degrees of freedom where n is the number of classes and m is the number of estimated parameters. The formula of this test is the following:

$$T = \sum_{i=1}^{n} \frac{(n_{observed,\ i} - n_{expected,\ i})^2}{n_{expected,\ i}}$$

The approach of the χ^2 distribution is better when $n_{expected,\ i} \geq 5$

Our null-hypothesis H_0 is that the data are uniformly distributed with significance level $a = 0.05$ and the alternative H_α that is not. After the implementation the value T of the χ^2-test is $T = 74.06 > \chi^2_{0.05,10} = 18.3$ so our null-hypothesis is rejected and we accept the alternative.

Another way to check our null-hypothesis and check if our data are uniformly distributed with a better power is the **Kolmogorov-Smirnov test**. In order to implement this test we use the formula:

$$D_n = \sup_x \{|F_n(x) - F(x)|\}$$

where F_n is the cumulative distribution function of the compared distribution and F is the cumulative distribution function of our data. The results of this test is $D = 0.0222 > D_{0.05,\,10000} = 0.0136$ which we obtain from the formula: $D = \frac{1.36}{\sqrt{n}}$ so we reject again our null-hypothesis.

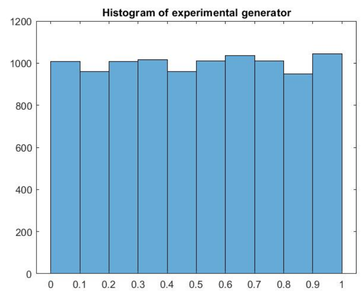
Additionally, we implemented the **Run-test I** to our data. This test is used mainly to compare the data with the median. The number of above and below runs is asymptotically distributed as:

$$\mathscr{N}(2\frac{n_1 n_2}{n_1 + n_2} + 1, 2\frac{n_1 n_2(2n_1 n_2 - n_1 - n_2)}{(n_1 + n_2)^2(n_1 + n_2 - 1)})$$

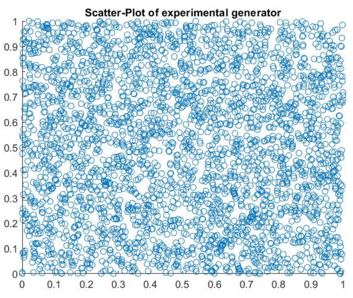
Finally, we had $z = 4.3636 > z_{0.05}$ so our data are not randomly distributed. After Run test I we evaluated the quality of our generator with **Run test II** where we had the result $Z = 191.5125 > x^2_{0.05,6}$ so our data set does not pass the Run test II. Finally, we tested our numbers with **Run-test III** which measures if the sequence of decreasing numbers is correlated with the sequence of increased numbers the result that we had implementing the formula $Z = \frac{X - \frac{2m-1}{3}}{\sqrt{\frac{16m-29}{90}}}$ where X is the total number of runs is $|Z| = 8.784022 > z_{0.05}$ so we reject our null hypothesis that the data are randomly distributed.

1.3 Experimenting with the Generator

In order to implement a generator which will allows us to accept the null hypothesis we changed the values a and M and this time we set $a = 601$ and $M = 100,001$. the results that we had are the following:



(a) Histogramme



(b) Scatter Plot

Figure 2

Statistical Tests

Our null hypothesis for $x^2 - test$ and K-S test is that the numbers are uniformly distributed.

x^2**-test**

The result was: $9.36 < x^2_{0.05,10} = 18.3$ so we accept our null hypothesis.

K-S-test

The D_{max} was: $0.007247 < D_{0.05,\,10000} = 0.0136$ so we accept our null hypothesis.

Our null hypothesis for Run tests I, II, III is that the numbers are randomly distributed.

Run test I

The result was: $|z| = 1.507 < z_{0.05}$ so we accept our null hypothesis.

Run test II

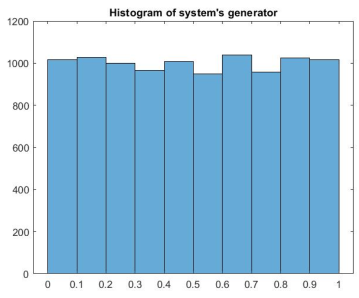
The result was: $Z = 3.3920 < x^2_{0.05,6}$ so we accept our null hypothesis.

Run test III

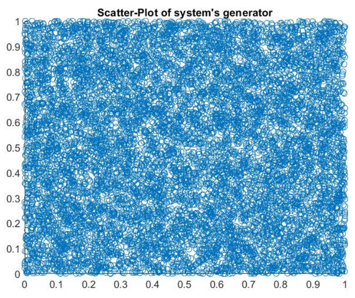
The result was: $|z| = 1.0831 < z_{0.05}$ so we accept our null hypothesis.

1.4 System's Generator

Using Matlab's random number generator('rand'). The numbers that we got are presented below



(a) Histogramme



(b) Scatter Plot

Figure 3

The comparison of the numbers that we obtained by system's function with the numbers of our generator is the following

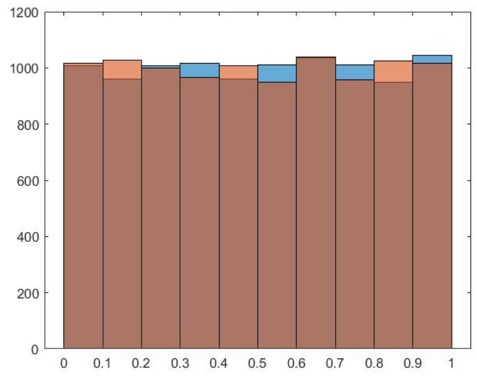


Figure 4: Histogramme

All the results that we had from the statistical tests drove us to accept the null Hypothesis. Specifically:

Statistical Tests

Our null hypothesis for x^2-test and K-S test is that the numbers are uniformly distributed.

x^2-test

The result was: $7.78 < x^2_{0.05,10} = 18.3$ so we accept our null hypothesis.

K-S-test

The D_{max} was: $0.007931 < D_{0.05,\,10000} = 0.0136$ so we accept our null hypothesis.

Our null hypothesis for Run tests I, II, III is that the numbers are randomly distributed.

Run test I

The result was: $|z| = 0.582347 < z_{0.05}$ so we accept our null hypothesis.
Run test II
The result was: $Z = 7.2237 < x^2_{0.05,6}$ so we accept our null hypothesis.
Run test III
The result was: $z = 0.245 < z_{0.05}$ so we accept our null hypothesis.

Finally, we tested all the previous generators by correlation test using the formula:

$$ch = \frac{1}{n-h} \sum_{i=1}^{n-h} U_i U_{i+h}$$

And the results were compared with N$(0.25, \frac{7}{144n})$ and we had:

1st generator	0.259
2nd generator	0.249
Matlab's generator	0.251

As we can see, our 1st generator returns a value quite far from the 0.25 for the small values of the standard deviation of the compared distribution. On the other hand, in the two last cases the result of the test is quite closer to 0.25.

2 Exercise 2. Discrete Random Variables

In this section we used Matlab's 'unifrnd' function to generate a sequence of 10,000 better uniformly distributed numbers. We will use those numbers in order to apply simulation methods and generate Discrete Random Numbers from some 'known' distributions. The numbers that we had are presented at the following plot.

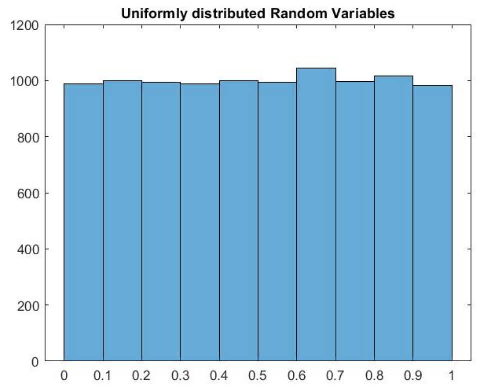


Figure 5: Histogramme

2.1 Geometric Distribution Simulation

Negative Binomial distribution $NB(k,p)$ is the sum of k independent Geometrical distributions $Geo(p)$. We set $k=1$ and $p=0.01$ in order to simulate the above 10,000 outcomes according to the following procedure.
$X = n$ if

$$F(n-1) < U \leq F(n) \Rightarrow 1-(1-p)^{n-1} < U \leq 1-(1-p)^n \Rightarrow n-1 < \frac{\log(1-U)}{\log(1-p)} \leq n$$

$$X = \left\lfloor \frac{\log(1-U)}{\log(1-p)} \right\rfloor + 1$$

The numbers we had after the implementation are presented below:

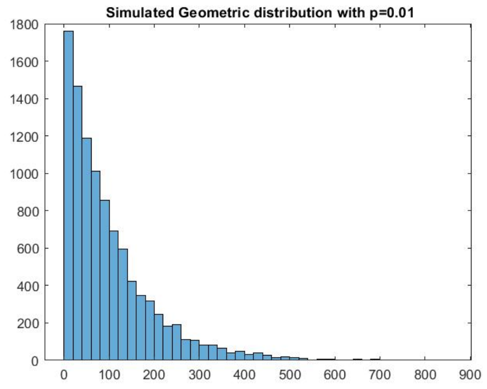


Figure 6: Geometrical Distribution

We tested our generator by x^2-test and with our null hypothesis to be that the numbers are geometrically distributed with $p=0.01$. Our null hypothesis was accepted with $p-value = 0.6288$. After the implementation of the above procedure we generated random numbers geometrically distributed with p=0.01 using system's generator. In the below plot we can compare the 2 sets of the random numbers.

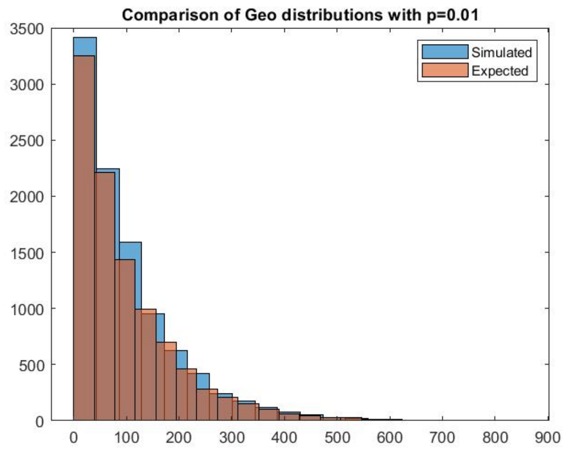


Figure 7: Geometrical Distributions

We tested matlab's generator by $x^2 - test$ and with our null hypothesis to be that the numbers are geometrically distributed with $p = 0.01$. Our null hypothesis was accepted with $p - value = 0.7045$.

2.2 6-Point Distribution

In this section we will simulate the 6 point distribution:

X	1	2	3	4	5	6
p_i	7/48	5/48	1/8	1/16	1/4	5/16

by applying a direct (crude) method, the rejection method and finally the Alias method.
Crude Method
Suppose X takes k distinct values $x_1 < x_2 < ... < x_k$ with $p_i = P(X = x_i)$, for $i = 1, 2, ..., k$. Then X takes the value x_i with probability p_i if U which is a random uniformly distributed number falls in an interval with length p_i. Specifically:

$$if \quad F(x_{i-1}) < U \leq F(x_i) \quad \Rightarrow \quad X = x_i$$

Rejection Method
Another method to simulate the above 6-point distribution is Rejection Method. This method is evolving following the below steps:

Assume $\text{P}(X = i) = p_i$, for $i = 1, 2, ..., k$. and Let $c \geq p_i$

1. $I = \lfloor (k * U_1) \rfloor + 1$

2. if $U_2 \leq p_I/c$ then output : I
3. Else go to step 1

Alias Method
The last method that we examined in this section is Alias method. This method is quite effective but its main drawback is that demands to evolve a complex set-up procedure. In order to implement this method we followed the below steps;

Set-up procedure

- Generate the table of $F(I)$-values,which part of the mass belongs to I itself).
- Generate the table of $L(I)$-values, (the alias of class I)

Method at Run time

1. Generate $I = \lfloor k * U_1 \rfloor + 1$
2. Generate U_2 uniformly distributed random number
3. If $U_2 \leq F(I)$ then return $X = I$ else return $X = L(I)$

In order to compare the results we had after the implementation of the above method we created the following bar chart which presents the relative frequencies of the different values.

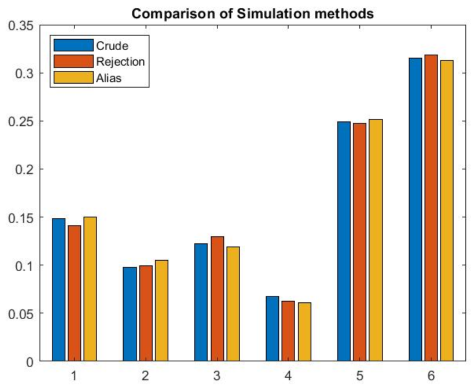


Figure 8: Geometrical Distribution

We tested our simulations by K-S test and null hypothesis that our numbers are distributed according to the given distribution. The result is that we accept our null hypothesis with $p - value = 1$. The simulation we did implementing the 3 methods gave us the numbers with relative frequencies quite close to the given frequencies. The results are presented below:

method — numbers	1	2	3	4	5	6
Given distr	7/48	5/48	1/8	1/16	1/4	5/16
Crude	0.1481	0.0978	0.1224	0.0671	0.2494	0.3152
Rejection	0.1413	0.0997	0.1299	0.0627	0.2476	0.3188
Alias	0.1502	0.1051	0.1191	0.061	0.2517	0.3129

3 Exercise 3

3.1 Exponential-Normal-Pareto distributions

In this section we generated simulated values from Exponential, Normal and Pareto distributions.

In order to generate these values from Exponential distribution we initially generated 10,000 uniformly distributed random numbers and then we used the following formula:

$$X = -\frac{\log(U)}{\lambda} \quad \sim \quad exp(\lambda)$$

We generated exponentially distributed numbers with $\lambda = 0.05$ and the values we had are the following

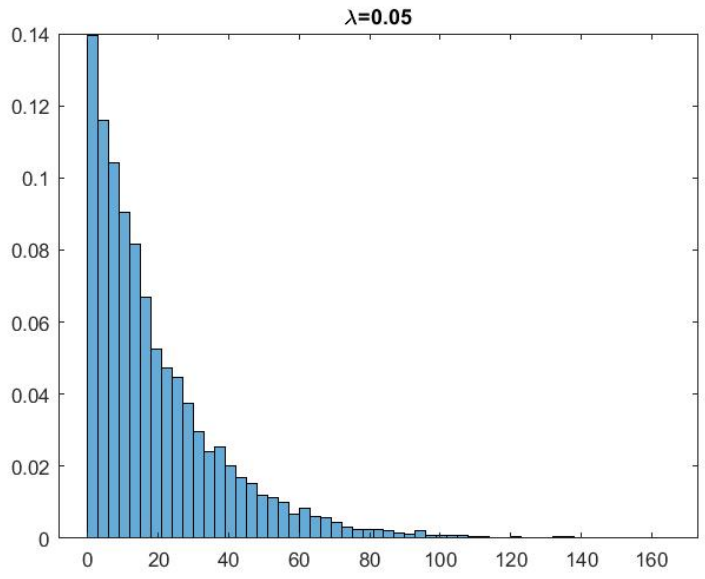


Figure 9: Exponential Distribution

We tested our set by K-S test with null hypothesis:"Numbers are exponentially distributed" and finally we accepted the null hypothesis with $p - value =$

0.8241
In order to generate random numbers from a Normal distribution we used Box-Muller method which gives us the disired values after a transformation from polar($\theta = 2\pi U_2, r = \sqrt{-2\log(U_1)}$) into Cartesian coordinates ($X = Z_1, Y = Z_2$) using the following formula:

$$\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = \sqrt{-2\log(U_1)}\begin{bmatrix} cos(2\pi U_2) \\ sin(2\pi U_2) \end{bmatrix}$$

The numbers we got after the implementation are presented below:

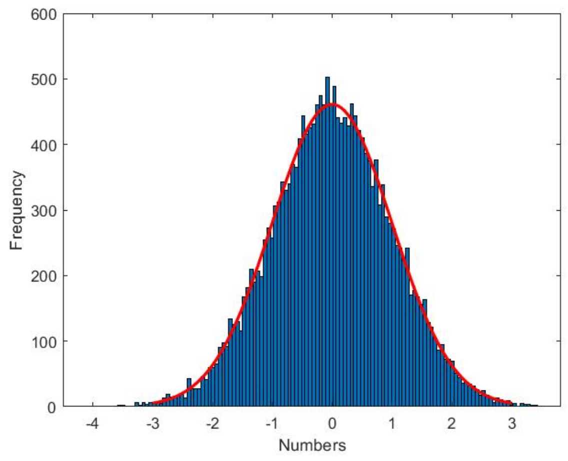


Figure 10: Normal Distribution

We tested our set by x^2-test with null hypothesis:"Numbers are normally distributed" and finally we accepted the null hypothesis with $p-value = 0.6583$
Finally, we simulated values from Pareto distribution with $\beta = 1$ for the different values $\kappa = 2.05, \kappa = 2.5, \kappa = 3, \kappa = 4$

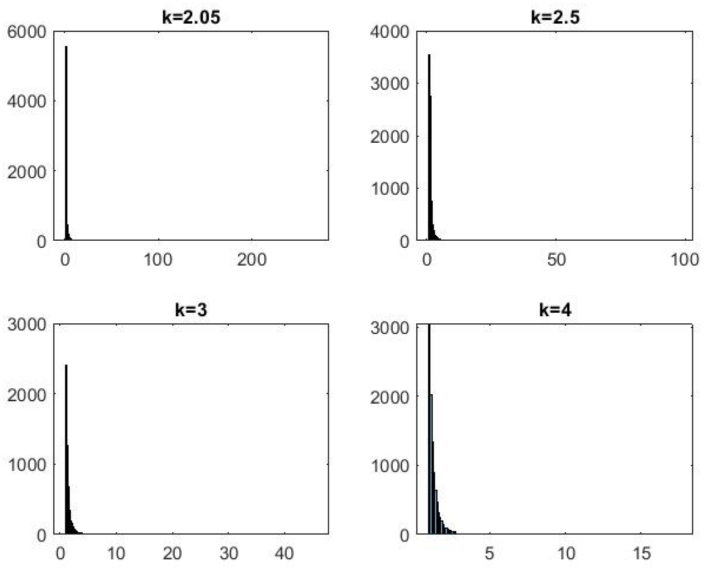


Figure 11: Pareto Distribution

For these numbers we used the following formulas to compare the actual mean value and variance with analytical results.

$$E(X) = \beta \frac{k}{k-1} \quad for\, k > 1$$

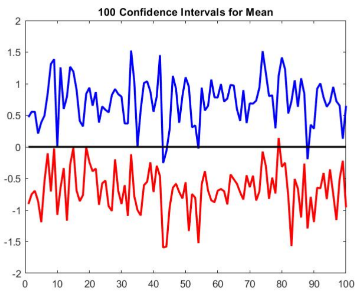
$$Var(X) = \beta^2 \frac{k}{(k-a)^2(k-2)} \quad for\, k > 2$$

k-values	Mean	Analytical Mean	Variance	Analytical Variance
k=2.05	1.9346	1.9524	4.7302	37.1882
k=2.5	1.6596	1.6667	1.4687	2.2222
k=3	1.4963	1.5000	0.6227	0.7500
k=4	1.3316	1.3333	0.2066	0.2222

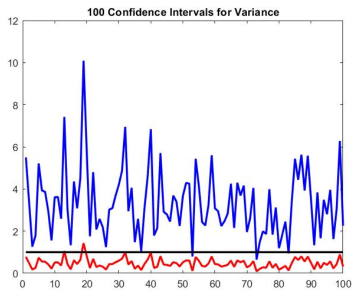
The results of the K-S tests with the null hypothesis H_0: "Numbers are Pareto distributed" are:

k-values	Accepted Hypothesis	$p-value$
k=2.05	H_0	0.3285
k=2.5	H_0	0.9666
k=3	H_0	0.6139
k=4	H_0	0.6139

In addition, for the normal distribution we generated 100 95% confidence intervals for mean and variance and the results are presented in the following plots:



(a) 100 C.I for Mean



(b) 100 C.I for Var

Figure 12

4 Exercise 4. Discrete Event Simulation

In this section we will create a event simulation of a blocking system. Suppose that we have 10 service units and this system is about to accept 10,000 customers where the time of arrivals following some specific distributions. The queue system will specify the time that each customer remains into the system when in the meantime new customers will arrive. Finally, the simulation that we will create will return us the proportion of the blocked customers. In order to obtain more safe results we will run the simulation 10 times. A general version of an algorithm that we used is specified below:

Algorithm 1 Discrete Event Simulation Algorithm

```
Require: $n$=number of servers, $m$=number of customers, $i$=customers passed from system, b = busy servers, r=rejected customers, t=time p=number of customers arrive

Initialize data
Set $t \leftarrow 0$
Create arrival and departure time of the first customer(8 Erlang)-(Exponential)
$b \leftarrow b + 1$
$i \leftarrow i + 1$
while $i < m$ do
  Define events
  if $t < t_{departure}$ then
    Event is arrival
    if p=0 then
      continue
    else
      for $j = 1, .., p$ do
        Create new customer times(Exponential)
        $i \leftarrow i + 1$
        if $b < n$ then
          Accept customer
          $b \leftarrow b + 1$
        else
          Reject customer
          $r \leftarrow r + 1$
        end if
      end for
    end if
  else
    Next event is departure
    $b \leftarrow b - 1$
  end if
end while
```

4.1 Poisson arrival process

In this section we modelled the arrival process as a Poisson process the service time following an exponential distribution with mean $m_1 = 8$ and mean time between customers $m_2 = 1$ corresponding to an offered traffic of 8 Erlang. Additionally we estimated a 95% confidence interval for the mean of the 10 iterations. The results we had are the following:

Blocked customers	Confidence Interval	Erlang Formula
0.1232	[0.1185,0.1278]	0.1217

4.2 Experimenting with Erlang distribution

In this section we changed the parameters of Erlang distribution so that the mean arrival times is 1. Specifically, we decreased the number of initial random variables at 4 and we set $\lambda = 4$ in order to have mean arrival times equal to 1. So we created an Erlang - 4 distribution with mean 1. We expect that the result will be even different compared to the previous case. The results we had were the following.

Blocked customers	Confidence Interval
0.1239	[0.1207,0.1272]

4.3 Hyper-exponential arrival times

We modelled again the above process, this time using hyper-exponential arrival times with parameters: $p_1 = 0.8, \lambda_1 = 0.8333, p_2 = 0.2, \lambda_2 = 5$. The results we had are the following.

Blocked customers	Confidence Interval
0.1328	[0.1292,0.1364]

4.4 Constant Service Time

We repeated again the above simulation using the same means for the arrival time corresponding to Erlang distribution as in a previous case but we set a constant service time equal to 8. We observe that in this case the result we had for the fraction of blocking customers is closer to the result we had from Erlang-B formula compared with 4.1 subsection where we used exponentially distributed service times. The results we had are presented below:

Blocked customers	Confidence Interval	Erlang-B Formula
0.1208	[0.1191,0.1225]	0.1217

4.5 Pareto distributed service time

In this section we modelled again the simulation corresponding to Pareto distributed service times with $k = 1.05$ and $k = 2.05$. Even if we used the same distribution the results we had were quite different.

k-value	Blocked customers	Confidence Interval
1.05	0.0056	[0.0030,0.0083]
2.05	0.1427	[0.1357, 0.1497]

4.6 Experimenting with Normal and Uniform distribution

Finally, we have chosen to use random variables from $X \sim N(1, 0.09)$ and $X \sim U(0, 2)$ to simulate the arrival times and keep mean equal to 1. The results we had are presented below:

Distribution	Blocked customers	Confidence Interval
$N(1, 0.09)$	0.1406	[0.1384,0.1428]
$U(0, 2)$	0.1541	[0.1511, 0.1572]

To sum up, we can observe that as the variance of the used distribution which corresponds to the different arrival or service times is increased, affects with an expansion to the limits of the confidence intervals.

5 Exercise 5

5.1 Crude Monte Carlo estimator

Some methods which can be used to find the solution of integrals is the Variance Reduction Methods. Using these methods we can make a simulation where we have smaller confidence intervals for the output. In this exercise we will estimate the integral $\int_0^1 e^x dx$. The above integral can be interpreted as

$$E(e^U) = \int_0^1 e^x dx, \quad U \in \mathbf{U}(0, 1)$$

In order to estimate the above integral we have to obtain a sample of the random variable e^U and take the average.

$$X_i = e^{U_i}, \qquad \bar{X} = \frac{\sum_{i=1}^n X_i}{n}$$

Finally,we used 100 sample and we got that $\int_0^1 e^x dx \approx 1.7315$ with a confidence interval [1.7215 , 1.7416] and $Var(X) = 0.2639$.

5.2 Antithetic Variables

The general idea of this method is to exploit the correlation. According to this method if the estimator is positively correlated with U_i then we use the antithetic variable to estimate the spaces on both sides of the function.

$$Y_i = \frac{e^{U_i} + e^{1-U_i}}{2}, \quad \bar{Y} = \frac{\sum_{i=1}^n Y_i}{n}$$

Implementing the above procedure using 100 samples again we had: $\int_0^1 e^x dx \approx 1.7252$ with a confidence interval [1.7239 , 1.7265] and $Var(X) = 0.0043$.

5.3 Control Variates

Suppose that we want to estimate $\theta = E[X]$ where X is the output of a simulation. Then, for any constant c we have that $X + c(Y - E[Y])$ is an unbiased estimator of θ where Y is another output with known expected value. In order to minimize the variance we set $c^* = -\frac{Con(X,Y)}{Var(Y)}$. So for the variance

we have $Var(X + c(Y - E[Y])) = Var(X) + c^{*2}Var(Y) + 2c^{*}Cov(X,Y) = Var(X) - \frac{Cov(X,Y)^2}{Var(Y)}$ where $X = e^{U_i}$ and $Y = U_i$.
We used the above method to estimate the integral $\int_0^1 e^x dx$ so that our simulated data have a smaller Variance and we had that $\int_0^1 e^x dx \approx 1.7242$ with a confidence interval [1.7229 , 1.7255] and $Var(X) = 0.0044$.

5.4 Stratified sampling

Using this method in order to calculate the above integral we apply a division of the interval in 99 pieces sampling in predetermined areas using the following formula:

$$W_i = \frac{e^{\frac{U_{i,1}}{100}} + e^{\frac{1}{100}+\frac{U_{i,2}}{100}} + ... + e^{\frac{99}{100}+\frac{U_{i,100}}{100}}}{100}$$

Implementing the above procedure using 100 samples again we had: $\int_0^1 e^x dx \approx 1.7215$ with a confidence interval [1.7205 , 1.7225] and $Var(X) = 0.0027$.

6 Exercise 6. Markov Chain Monte Carlo Simulation

In this section we will implement a Random Walk Metropolis Hastings algorithm for a trunk group. Our distribution is given by

$$f(x) = cg(x)$$

where even if the unormalized density g can be evaluated easily, it is not so easy to evaluate:

$$c = \frac{1}{\int_X g(x)dx}$$

In order to implement Metropolis-Hastings distribution we have to follow the next steps:

- Propose $h(x, y)$
- Calculate the probability $min(1, \frac{f(y)h(y,x)}{f(x)h(x,y)}) = min(1, \frac{g(y)h(y,x)}{g(x)h(x,y)})$
- Implement the random walk algorithm according to the next steps
  1. Sample from Metropolis-Hastings function $f(x)$
  2. At iteration i the state is X_i
  3. Propose to jump from X_i to $Y_i = X_i + \Delta X_i$ where ΔX_i is sampled from a symmetric distribution
  4. If $g(Y) \geq g(X_i)$ accept
  5. If $g(Y) \geq g(X_i)$ accept the weighted probability$\frac{g(Y)}{g(X_i)}$

6. On accept: Set $X_{i+1} = Y_i$ and repeat
7. On reject: Set $X_{i+1} = X_i$ and repeat

The results that we had implementing the above algorithm to generate values from distribution $P(i) = \frac{\frac{A^i}{i!}}{\sum_{j=0}^{m} \frac{A^j}{j!}}$ as is presented at the following plot.

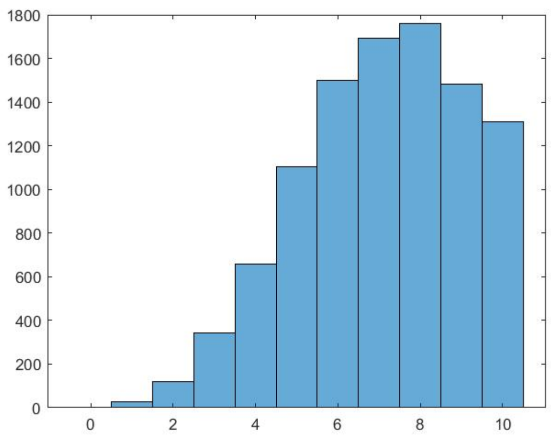


Figure 13: Histogramme

In addition we implemented again the above algorithm for the 2-dimentional case of

$$P(i,j) = \frac{A_1^i A_2^j}{K i! j!}, \quad 0 \leq i + j \leq n$$

using $A_1 = A_2 = 4$ and n=10 with our proposed distribution to be the uniform distribution. The following plot represents our results.

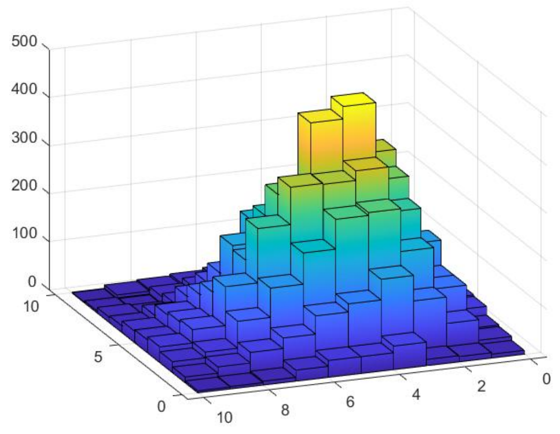


Figure 14: Histogramme

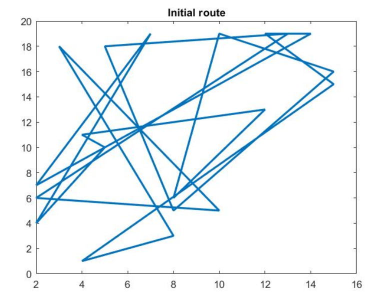
7 Simulated Annealing

7.1 "Euclidean" Cost

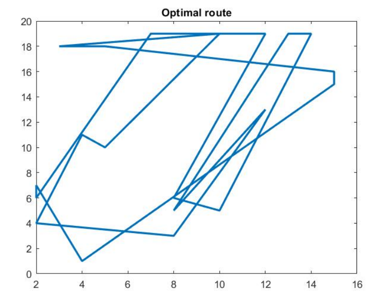
In this Part we will solve Travelling Salesman Problem and using a map with distances between some cities we will try to find a permutation of the cities in order to have the minimum cost. Firstly, we will create 20 random points with coordinates from the interval [1,20] and we assume that the cost of travelling from one city to another is the Euclidean distance between the two cities. The cost is represented by the following function:

$$f(c, d_{i,j}) = \sum_{i=1}^{n-1} d_{c_i,c_{i+1}} + d_{c_n,c_1}$$

We use as temperature the decreasing function $T_k = \frac{1}{\sqrt{1+k}}$ where k is the number of iterations. At each step we update the state according to the random walk Metropolis-Hastings algorithm. The whole procedure is based on an initial permutation of the state X_i in order to generate a candidate Y_i. If the new candidate has a lower cost than the old state we accept the change. Otherwise, we accept only with probability : $e^{\frac{-(U(Y_i)-U((X_i))}{T_i}}$ for a symmetric proposal distribution. In the following plots we can see the difference between the initial and the final route.



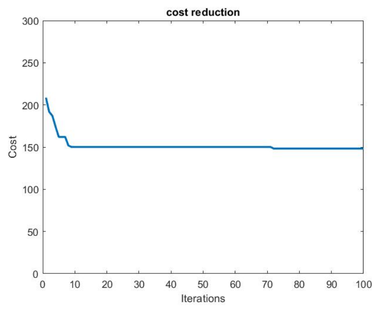
(a) Initial Route



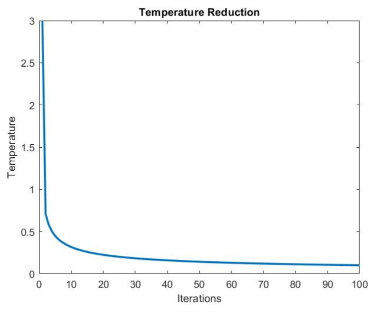
(b) Final Route

Figure 15

Our initial cost of the whole route was 208.58 and finally the cost reduced at 148.3. Below, the plots represent the reduction of the cost and the temperature.



(a) Cost

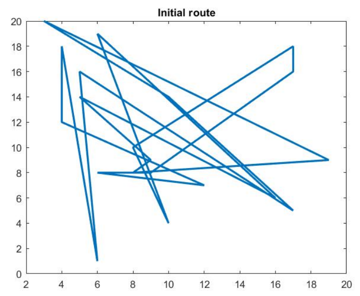


(b) Temperature

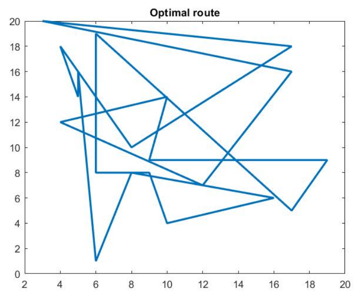
Figure 16

7.2 Using Cost Matrix

We implemented again the above procedure this time using a cost matrix which every column represents the cost of going from city $i \to j$. The following plots we can see the initial and the final route without the distance affecting the total cost. The points we provided were only to visualise the travel.



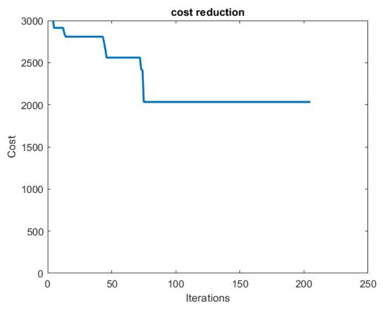
(a) Inital Route



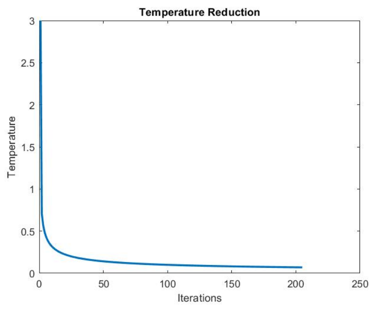
(b) Final Route

Figure 17

The initial cost was 3705.0 and finally our final cost is 2033.0. Below we can see the reduction of the cost and the temperature.



(a) Cost



(b) Temperature

Figure 18

8 Exercise 8

8.1 Bootstrap Method

One of the most significant statistical techniques for estimating quantities such as the mean of a population is bootstrap method. Using this method we can estimate the desired quantities by averaging estimates from simulating samples. The general method is described below:

1. Given a data set with n observations we have to choose and simulate r data sets.
2. Each of the simulated dataset should has n observations sampled from the empirical distribution.

3. In order to simulate each of those data sets we simply have to choose samples from the original data set with replacement.
4. Find a bootstrap replicate estimating the desired parameter for each of the simulated data sets.
5. Finally, we compute the parameter of interest using the bootstrap replicate.

Let $X_1, X_2, \ldots.., X_n$ be independent and identically distributed random variables having unknown mean μ. In order to estimate the probability $p = P\{\alpha < \sum_{i=1}^n X_i/n - \mu < \beta\}$ we used the above method to find the bootstrap replicate and determine the theoretical means and finally we computed the probability the quantity $\sum_{i=1}^n X_i/n - \mu$ to be in the given range.

In addition we implemented the above procedure to estimate p if $X_i = \{56, 101, 78, 67, 93, 87, 64, 72, 80, 69\}$, $\alpha = -5$ and $\beta = 5$ and we found $p = 0.8$.

8.2 Bootstrap estimate of Var(S^2)

Let $X_i = \{5, 4, 9, 6, 21, 17, 11, 20, 7, 10, 21, 15, 13, 16, 8\}$ be a sample from a distribution with unknown variance σ^2. Using bootstrap method to estimate σ^2 we chose 15 samples with replacement and we computed $S^2 = \sum_{i=1}^n (X_i - \bar{X})^2/(n-1)$ and finally we estimated $Var(S^2) = 54.85$

8.3 Experimenting with Pareto distributed random numbers

In this section we generated N=200 Pareto distributed random numbers with $\beta = 1$ and $k = 1.05$ using the function that we made in a previous section. The mean was 5.55 and the median 1.96. We used bootstrap method again but this time to estimate the variance of the sample mean and the variance of the sample median and we had:

$$Var(\bar{X}) = 1.47, \qquad Var(median) = 0.022$$

The difference between the two variances is due to the fact that the median is not affected from the extreme values of a sample. Finally in the table below you can find the estimated mean and the median as well as the same quantities for the population.

Quantity	Estimated	Population's
Mean	5.50	5.55
Median	1.94	1.96