Technical University of Denmark

02443 Stochastic Simulation

Report of Exercises

Name:
Sanaz Behboodi
Jie Xu

Student # :
s180176
s181238

June 19, 2019

1 Exercise 1: Random number generation

The aim of this exercise is generating pseudo random numbers by a computer algorithm such that the output is a sequence of reals or integers, which suppose to be uniformly distributed on $[0, 1]$ and statistically independent.

1.1 Linear Congruential Generator (LCG)

In the first step of this exercise, Linear Congruential Generator (LCG) is implemented for generating 10000 random numbers by

$$x_i = mod(ax_{i-1} + c, M), U_i = \frac{x_i}{M} \tag{1}$$

Where a is a multiplier, c is a shift and M is modulus, then these numbers is presented by a histogram for 10 classes.

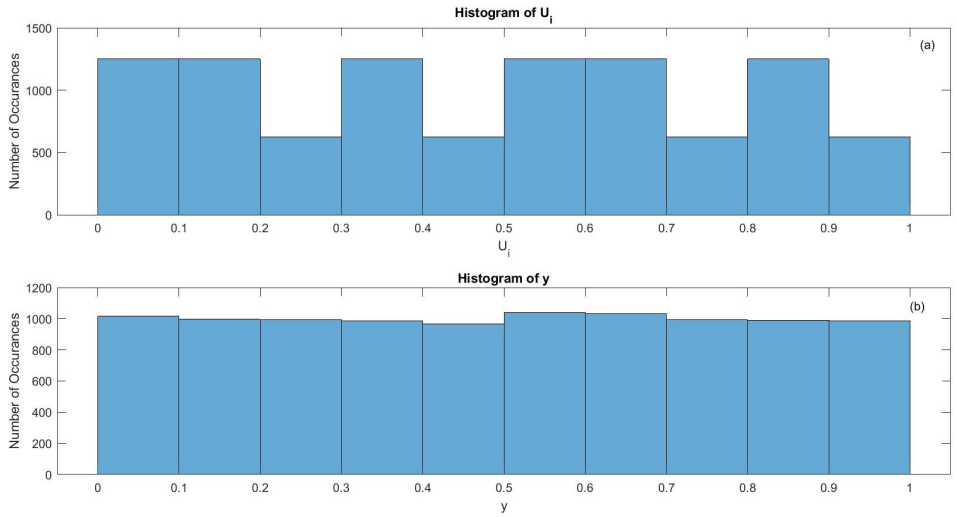

Figure 1: Histogram for U and Y.

Figure 1 show the histogram for U which generated by LCG and the histogram for Uniformly distributed random numbers.

1.2 Evaluate the quality of the generators

The quality of the generators is evaluated by graphical descriptive statistics (histogrammes, scatter plots) and statistical tests (X^2,Kolmogorov-Smirnov, run-tests, and correlation test).

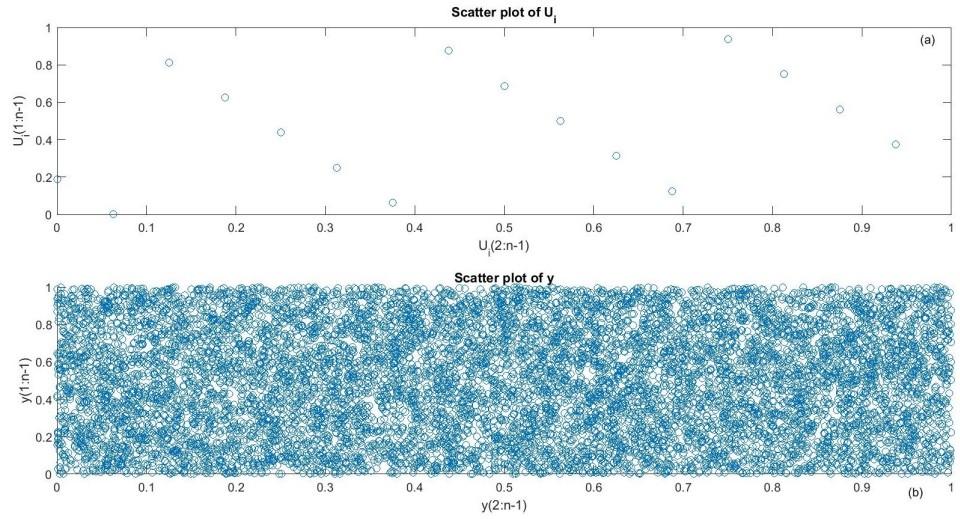

Figure 2: Scatter plot for U and Y.

In this part of exercise, A chi-squared test (χ^2 test) as a statistical hypothesis test is implemented that is given by

$$T = \Sigma_{n_{classes}}^{i=1} = \frac{(n_{observed,i} - n_{expected,i})^2}{n_{expected,i}} \tag{2}$$

The chi-squared test is used for the randomness of data to determine whether there is a significant difference between the expected frequencies and the observed frequencies in one or more categories.A chi-squared test can be used to attempt rejection of the null hypothesis that the data are independent.

Now for testing the type of distribution of the random number generator, we implemented Kolmogorov-Smirnov test that can be used to compare a sample with a reference probability distribution. In this respect, the Kolmogorov–Smirnov statistic compares empirical distribution function $F_n(x)$ with hypothesized distribution $F(x)$ and it quantifies a distance that is given by

$$D_n = sup_x|F_n(x) - F(x)| \tag{3}$$

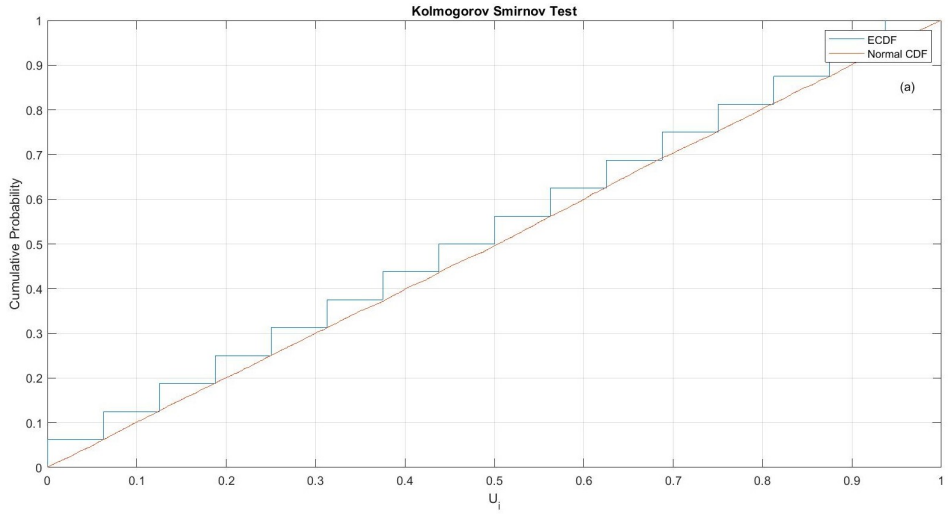

Figure 3: Kolmogorov–Smirnov Test .

Figure 3 shows the Kolmogorov–Smirnov (KS) test.
Run test is other test which examines the arrangement of numbers in a sequence to test the hypothesis of independence and it can be used by comparing with the median.Let n_1 and n_2 be the number of individual observations above and below the mean, let b the total number of runs then the mean and variance of b can be expressed as

$$\mu_b = \frac{2n_1n_2}{2n_1 + n_2} + 1 \tag{4}$$

$$\sigma_b^2 = \frac{2n_1n_2(2n_1n_2 - n_1 - n_2}{(n_1 + n_2)^2(n_1 + n_2 - 1)} \tag{5}$$

b is approximately normally distributed .
Correlation test is another independence test is given by

$$C_h = \frac{1}{n-h}\Sigma_{n-h}^{i=1}U_iU_{i+h} \sim N(0.25, \frac{7}{144n}) \tag{6}$$

2 Exercise 2: Generation of random variables-Discrete sample space

The aim of this exercise is to generate the independent, discrete random variable X_i having probability mass function

$$P\{X = x_j\} = p_j, j = 0, 1, ..., \Sigma_j p_j = 1 \tag{7}$$

To accomplish this, we assume, we have access to random number U_i which is uniformly distributed over $(0, 1)$ and finally transform U_i to X_i. In this exercise we generated uniform random variables and we compared the results obtained in simulation with expected results with using histograms and tests.

2.1 Geometric distribution

first part of the exercise, Geometric distribution which is given by

$$F(n) = P(X \leq n) = 1 - (1-p)^n \tag{8}$$

Where $X = \lfloor(\frac{\log(U)}{\log(1-p)})\rfloor + 1$ with value of $p = 0.1$ is implemented and simulated for 10000 outcomes that shows in figure 4.

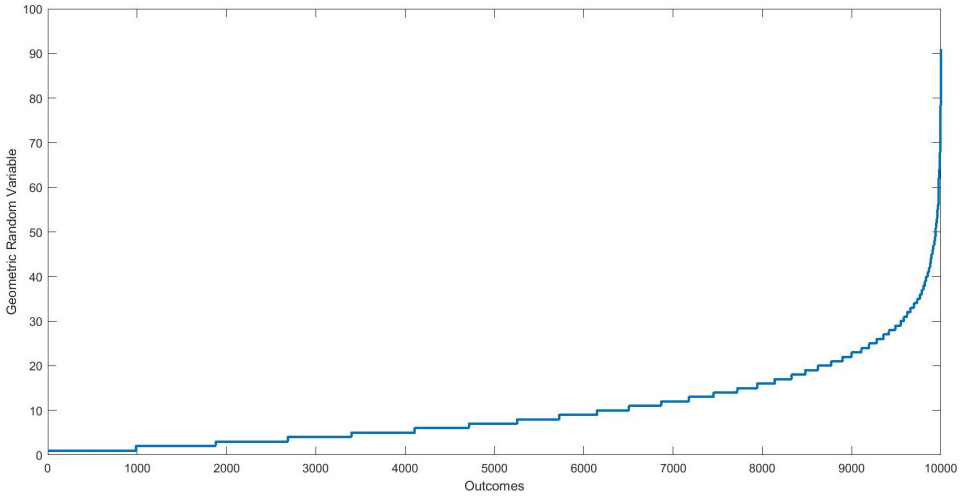

Figure 4:Geometric random variables for 10000 outcomes .

2.2 Simulate the 6 point distribution

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

Table 1: 6-point distribution.

Figure 5 shows generating random variables and their histograms by By applying a direct(crude) method, rejection method and Alias method.

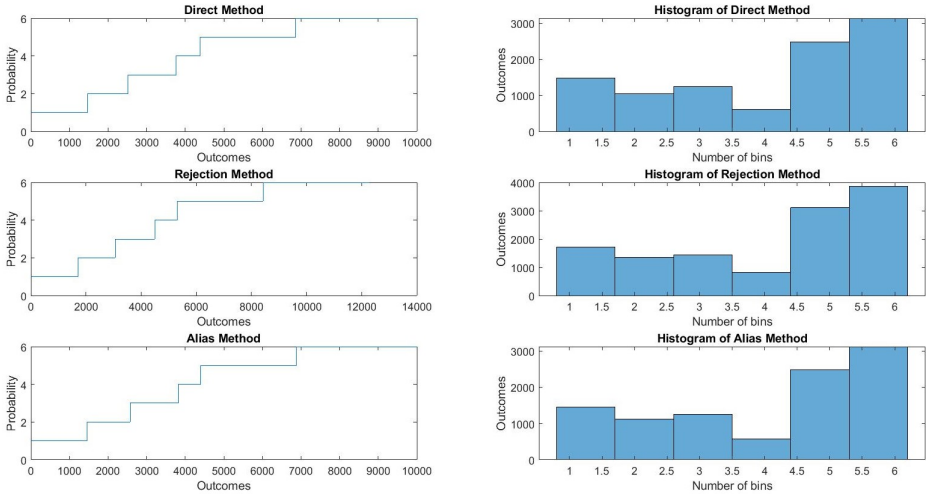

Figure 5: Different method for generating random variables and their histograms .

Table2 presents result of testing of generated uniform random variables with different methods .

Test	Crude Method	Rejection Method	Alias Method
χ^2 Test	1	1	1
KS Test	1	1	1
Run Test	0	1	1
CC Test	1	1	1

Table 2: Testing of generated uniform random variables with different method .

3 Exercise 3: Generation of random variables-Continuous sample space

As same as last exercise the aim of this exercise is to generate the independent and continuous random variable by using the inverse transform approach and the rejection approach.

3.1 Generated simulated value

The first part of the exercise we generated simulated value from following distributions

1: Exponential distribution:

Let $X \sim \exp \lambda$ according inversion method, we have $X = F^{-1}(U)$
If $F(x) = 1 - \exp(\lambda x)$, then $F^{-1}(u) = -\frac{1}{\lambda}\log(1-u)$ (both 1-U and U are uniformly distributed.

$$X = -\frac{\log(U)}{\lambda} \sim \exp(\lambda) \tag{9}$$

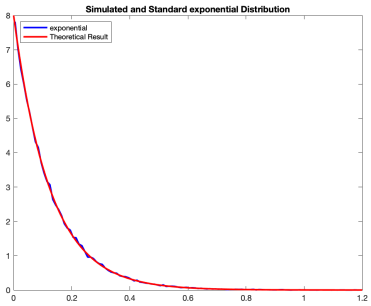

Figure 6: simulated values from Exponential distribution.

2: Normal distribution (at least with standard Box-Mueller):
Let $Z_1, Z_2 \sim N(0,1)$,the polar method or Box-Mueller is a way that transform from polar coordinate $\theta = 2_2, r = \sqrt{-2\log(U_1)}$ into Cartesian coordinates$X = Z_1$ and $Y = Z_2$ which is given by

$$Z_1 = r\cos\theta \tag{10}$$

$$Z_2 = r\sin\theta \tag{11}$$

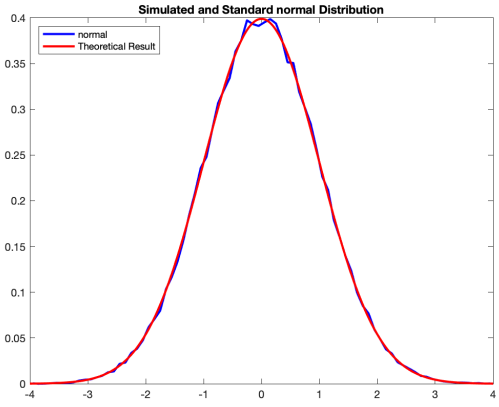

Figure 7: simulated values from Normal distribution.

3:Pareto distribution, with $\beta = 1$ and different values of $K = 2.05, 2.5, 3, 4$

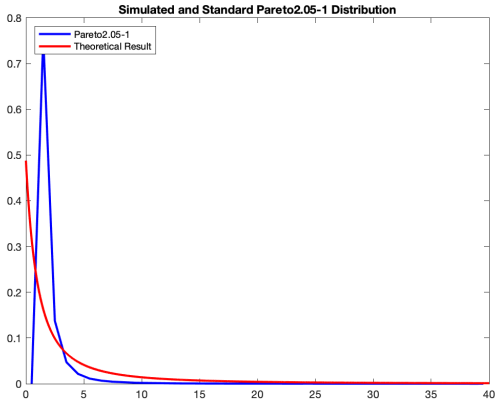

Figure 8: simulated values from Pareto distribution for K=2.05-1.

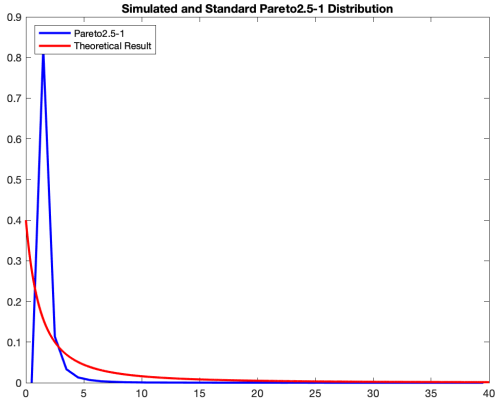

Figure 9: simulated values from Pareto distribution for K=2.5-1.

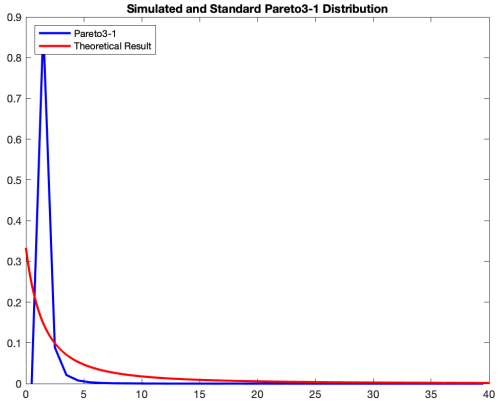

Figure 10: simulated values from Pareto distribution for K=3-1.

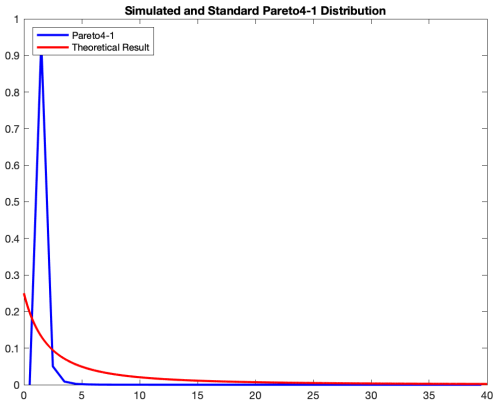

Figure 11: simulated values from Pareto distribution for K= 4-1.

Distribution	Mean	lbMean	ubMean	Median	Variance	lbMean	ubMean	X2 Test	KS tests
exp(8)	0.125258 (0.125)	0.125162	0.125355	0.086909 (0.086643)	0.015594 (0.015625)	0.015526	0.015662	Reject	Accept
standard norm	0.004697 (0)	-0.001485	0.010879	0.004514 (0)	0.997409 (1)	0.993057	1.001800	Reject	Accept
Pareto(2.05, 1)	1.947476 (1.952381)	1.908489	1.986464	1.403764 (1.402310)	6.290309 (37.188209)	6.317999	6.262862	Reject	Reject
Pareto(2.5, 1)	1.665760 (1.666667)	1.655095	1.676424	1.320630 (1.319508)	1.720617 (2.222222)	1.713109	1.728191	Reject	Reject
Pareto(3, 1)	1.500061 (1.500000)	1.495828	1.504294	1.260814 (1.259921)	0.682951 (0.750000)	0.679971	0.685957	Reject	Reject
Pareto(4, 1)	1.333713 (1.333333)	1.332379	1.335047	1.189839 (1.189207)	0.215267 (0.222222)	0.214327	0.216214	Reject	Reject

Table 3: Analysis and Test of the Simulation Result.

4 Exercise 4: Discrete simulation/event-by-event

In this exercise we write discrete event simulation program for a blocking system,with n service units and no waiting room.
First, the arrival process is modelled as a Poisson process and the service time distribution is modeled as exponential for $n = 10$(number of service units), mean service time = 8 time units, mean time between customers = 1 time unit and 10.000 customers, then record the fraction of blocked customers, and a confidence interval for this fraction.
In the second part of this exercise, we substitute the arrival process with a renewal process with
1:Erlang distributed inter arrival times mean of 1
2:hyper exponential inter arrival times with $p_1 = 0.8$, $\lambda_1 = 0.8333$, $p_2 = 0.2$, $\lambda_2 = 5.0$.
Finally experiment with different service time distributions. Suggestions are constant service time and Pareto distributed service times with k = 1.05 and k = 2.05.
All this results is presented in Table 4.

Id	Inter-Arrival Time	Service Time	Theoretically	Mean	Variance	LB	UB	Ela-Time
1	exp(1)	exp(8)	0.121661	0.121679	0.000040	0.121428	0.121930	23.269 s
2	Erlang(1)	exp(8)	-	0.07743	0.000023	0.077425	0.077435	36.285 s
3	he(0.8-0.833, 0.2-5.0)	exp(8)	-	0.000143	2.6718e-08	0.00014299	0.00014301	21.884 s
4	exp(1)	cons(8)	0.121661	0.12187	0.000020	0.12186	0.12187	15.135 s
5	exp(1)	Pareto(1.05, 0.38)	0.120757	0.001418	0.000006	0.0014166	0.0014194	16.210 s
6	exp(1)	Pareto(2.05, 4.10)	0.121876	0.001277	0.000003	0.0012762	0.0012778	17.511 s

Table 4 :Discrete Event Simulation of Blocking System .

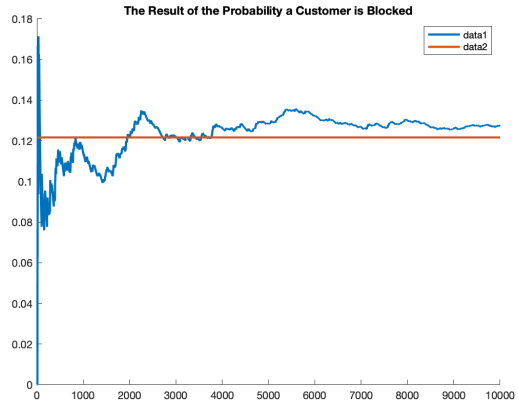

Figure 12: The result of the probability a customer is blocked.

5 Exercise 5: Variance reductions method

Variance reduction is a procedure used to increase the precision of the estimates that can be obtained for a given simulation, in order to make a simulation statistically efficient and obtain a greater precision and smaller confidence intervals for the output random variable of interest, variance reduction techniques can be used.The aim of this exercise is using some methods such as Antithetic variables, Control variates, Stratified sampling and importance sampling for variance reduction.

5.1 The crude Monte Carlo estimator

We want to estimate $\int_0^1 e^x dx$ by using crude Monte Carlo estimator based on 100 samples and present the result as the point estimator and a confidence interval.
This integral can be interpreted as

$$E(e^U) = \int_0^1 e^x dx = \theta, U \in U(0,1) \tag{12}$$

TO estimate the integral,we sample of the random variable e^U ($x_i = e_i^U$ and we take the average,

$$\overline{X} = \frac{\Sigma_n^{i=1} X_i}{n} \tag{13}$$

5.2 Antithetic variables

As mentioned before, doing simulation for estimating parameter $\theta = \overline{X}$ is more efficient if $Var(X)$ is reduced. Let identically distributed random variables X_1 and X_2 is generated then

$$Var(\frac{X_1 + X_2}{2}) = \frac{1}{4}[var(X_1) + var(X_2) + 2Cov(X_1, X_2)] \tag{14}$$

Variance is reduced if X_1 and X_2 have $Cov(X_1, X_2) \leq 0$. We want to estimate $\int_0^1 e^x dx$ by using Antithetic variables

5.3 Stratified sampling

Stratified random sampling is a method of sampling that involves the division of a population into smaller sub-groups known as strata. In Stratified sampling members, the strata are formed based on shared attributes or characteristics such as income or educational attainment.

Table 5 shows the estimation of the integral for 3 different methods which is presented in sections 5.1, 5.2 and 5.3.

5.4 Control variates

In this part of exercise we used control variates to reduce the variance of the estimator in exercise 4 (Poisson arrivals).
Table 6 shows control variates which reduce the variance of the estimator in exercise 4.

5.5 Simulation Result

Method	nSample	Mean	Variance	Lower Bound	Upper Bound	Elapsed Time
crude	10000	1.722460	0.244188	1.712773	1.732146	0.011881 s
antithetic	10000	1.718064	0.003885	1.716842	1.719286	0.005309 s
control	10000	1.717911	0.003901	1.716687	1.719136	0.007101 s
stratified	10000	1.718110	0.000263	1.718428	1.717792	0.008605 s
theoretical	-	(1.718282)	0.242036	-	-	-

Table 5: Estimation of the integral by simulation. Present the point estimator and confidence interval.

Method	Mean	Variance	Lower Bound	Upper Bound	Elapsed Time
exp/exp	0.121679	0.000040	0.121428	0.121930	23.269757 s
analytical result	0.121661	-	-	-	-
e/e control variates	0.121705	0.000025	0.121504	0.121905	-

Table 6:Control Variates in Exercise 4.

6 Exercise 6: Markov Chain Monte Carlo

the aim of this project is simulating the value of a random vector X whose component random variables are dependent. Markov chain Monte Carlo (MCMC) method is a powerful approach for generating a vector whose distribution is approximately of X. MCMC comprise a class of algorithms for sampling from a probability distribution. By constructing a Markov chain that has the desired distribution as its equilibrium distribution, one can obtain a sample of the desired distribution by observing the chain after a number of steps. The more steps there are, the more closely the distribution of the sample matches the actual desired distribution.

6.1 Metropolis-Hastings algorithm

Metropolis–Hastings algorithm is a Markov chain Monte Carlo (MCMC) method for obtaining a sequence of random samples from a probability distribution from which direct sampling is difficult. This sequence can be used to approximate the distribution.

First part of this exercise, we generate Random Walk Metropolis-Hastings (RW-MH) Simulation of the Queue Problem.

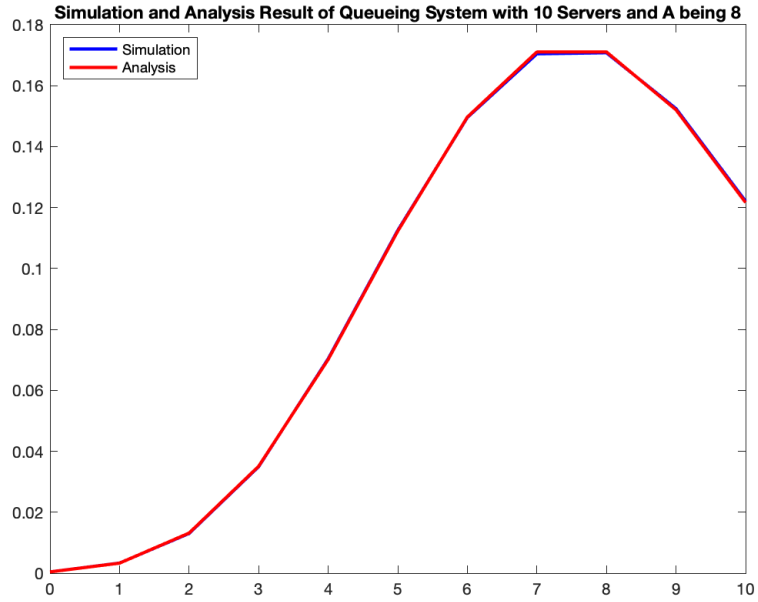

Figure 13: Simulation and analysis result of queuing system with 10 servers and A being 8.

6.2 Metropolis-hasting with two dimensional

we want to use Metropolis-Hastings, directly and coordinate wise to generate variates from this distribution. we use $A1, A2 = 4$ and $n = 10$.

Figure 11 shows simulation and analysis result of generating values by Metropolis-Hastings algorithm in one dimensional.

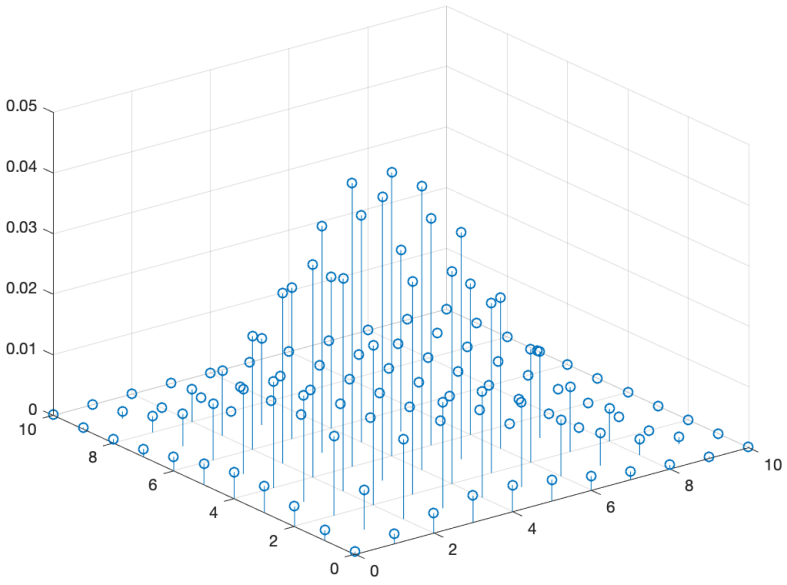

Figure 14: Analytical Result of Two-Dimension Problem.

A algorithm for direct Metropolis-Hastings of two-dimension problem is implemented, which can be called "Array the 2-D Irregular Sample Space for 8-Direction Random Walk". The sample space like the following one is caused by the external conditions. So we have to find all the combinations in the sample space, and sort them randomly:

(0, 3)
(0, 2) (1, 2)

(0, 1) (1, 1) (2, 1)
(0, 0) (1, 0) (2, 0) (3, 0)

(0, 2) (0, 3) (2, 0) (3, 0) (1, 1) (0, 0) (2, 1) (1, 0) (1, 2) (0, 1)

Then, we array the sample space to rectangle, where we can perform two-dimension random walk. The head and tail of every column, row are connected respectively, so the random walk will not exceed the boundary:

(0, 2) (0, 3) (2, 0) (3, 0) (1, 1)
(0, 0) (2, 1) (1, 0) (1, 2) (0, 1)

The above illustration is a simple one, while this problem has 66 combinations in total in sample space. The effect of the random walk is shown in the following figure.

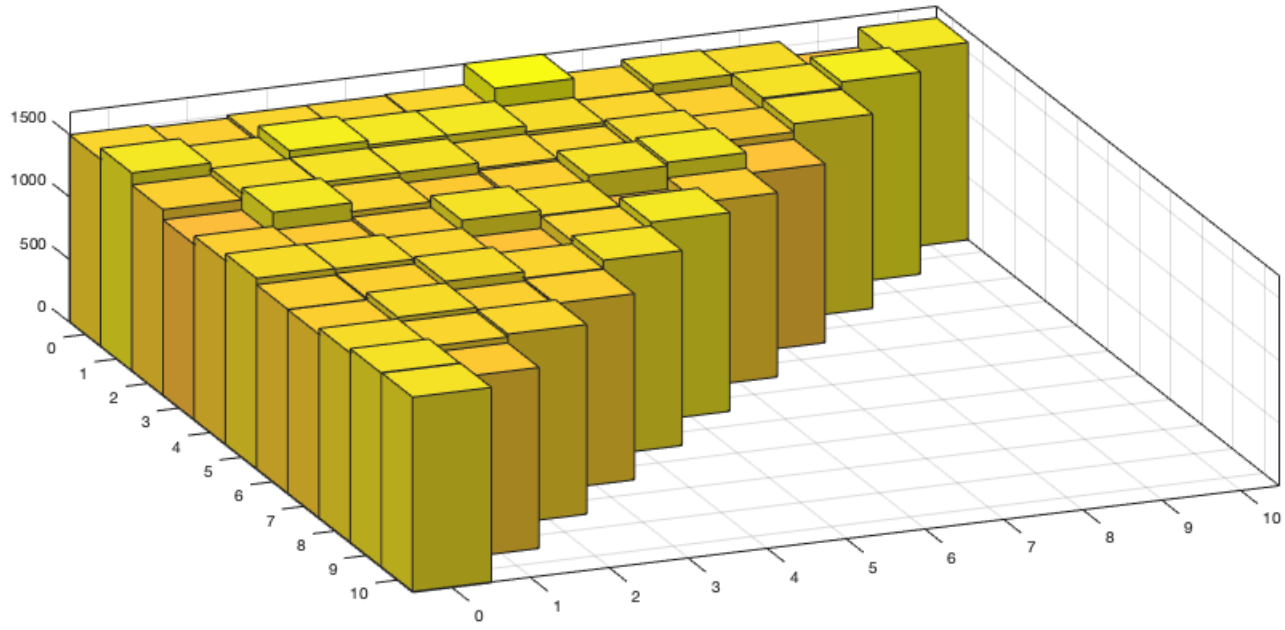

Figure 14: Direct Metropolis-Hastings of Two-Dimension Problem.

Figures 15 and 16 present the generation of values by Metropolis-Hastings, directly and coordinate wise respectively. The red point the theoretical value. It can be seen that the algorithm works very well, and the statistical test result is shown in table 7.

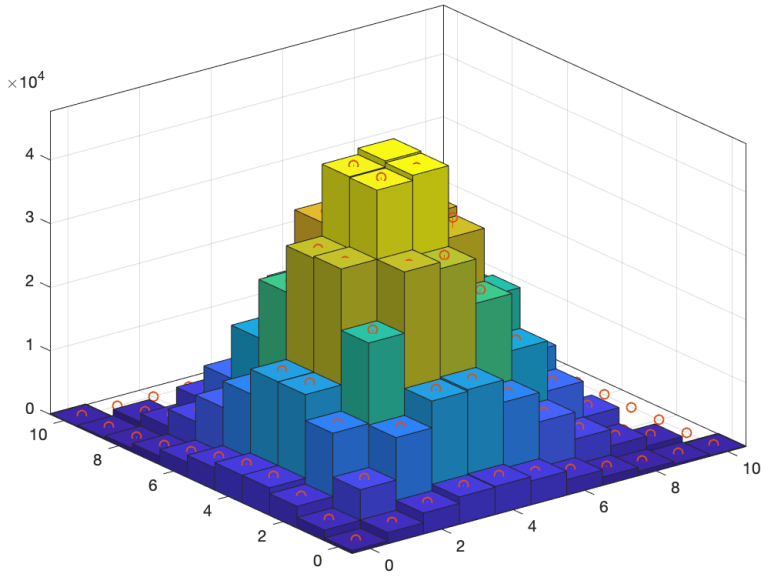

Figure 15: Direct Metropolis-Hastings of Two-Dimension Problem.

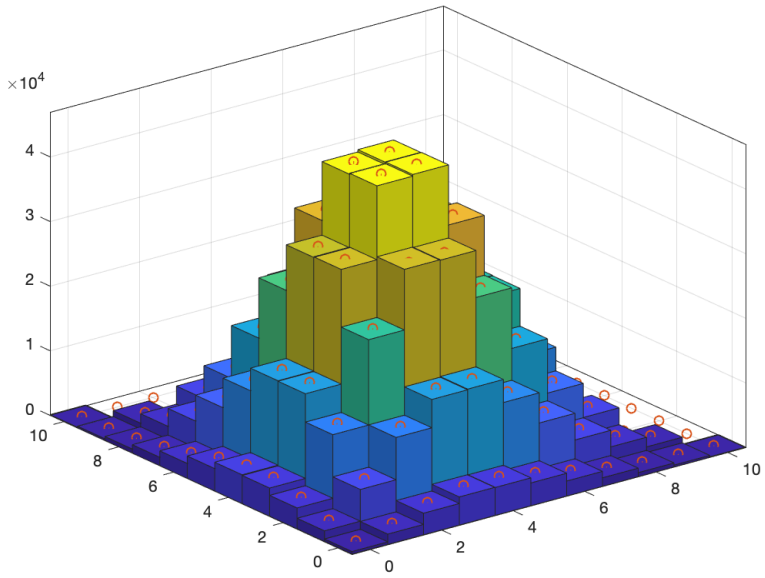

Figure 16: Coordinate-Wise Metropolis-Hastings of Two-Dimension Problem.

6.3 Gibbs sampling

Gibbs sampling is a Markov chain Monte Carlo (MCMC) algorithm for obtaining a sequence of observations which are approximated from a specified multivariate probability distribution, when direct sampling is difficult. This sequence can be used to approximate the joint distribution or the marginal distribution of one of the variables.
The Gibbs sampling assumes for any $i, i = 1, ..., n$ and any values x_j, $j \neq i$, we can generate random variable X having the probability mass function

$$P\{X = x\} = P\{X_i = x | X_j = x_j, j \neq i\} \tag{15}$$

$$P(i,j) = \frac{1}{K}\frac{A_1^i}{i!}\frac{A_2^j}{j!} = \frac{g(i,j)}{K} \qquad 0 \geq i+j \geq 10 \tag{16}$$

$$P\{X = x\} = P\{X_i = x | X_j = x_j, j \neq i\} = \frac{P\{X_i = x, X_j = x_j, j \neq i\}}{P\{X_j = x_j, j \neq i\}} \tag{17}$$

$$P\{X = x\} = \frac{g(x, x_j)}{\sum_{k=0}^{10-x_j} g(k, x_j)} = \frac{A_1^x}{x!} L_{x_j} \qquad L_{x_j} = \frac{A_2^{x_j}}{x_j! \sum_{k=0}^{10-x_j} g(k, x_j)} \tag{18}$$

Figure 17 shows coordinate wise solution using Gibbs sampling.

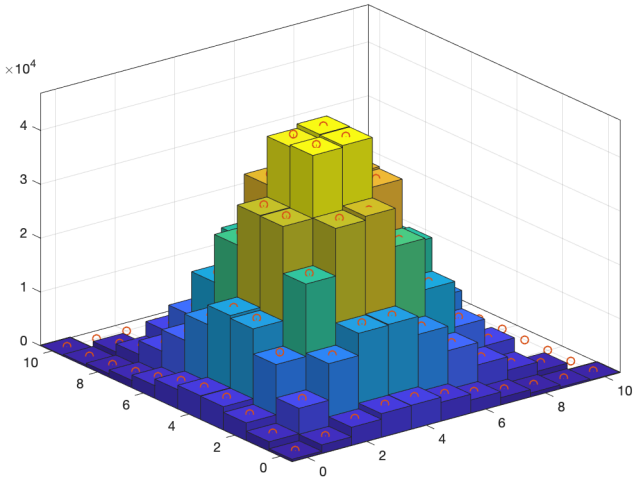

Figure 17:Coordinate wise Gibbs sampling .

Table 5 show χ^2 values and Critical χ^2 values for 4 different methods.

ID	Dim	Method	nSample	Chi-Square	Critical Chi-Square	Reject
1	1	Direct	1000000	8.603061	18.307038 (0.05)	false
2	2	Direct	1000000	243.148627	84.820645 (0.05)	true
3	2	Gibbs Sampler	1000000	71.384375	84.820645 (0.05)	false
4	2	Coordinate-Wise	1000000	239.480581	84.820645 (0.05)	true

Table 7:results of χ^2 Test for different methods .

7 Exercise 7: Simulated annealing

Simulated annealing is a probabilistic technique for approximating the global optimum of a given function, in other word,it attempts to find the global optimum in presence of multiple local optima.
The aim of this exercise is finding a route S (a permutation of $1, ..., n$) with minimal total cost $\Sigma_{n_1}^{i=1} A(S_i, S_{i+1})$ in Travelling salesman problem (TSP) with Given n stations, and an n-by-n matrix A giving the cost of going from station i to j, which starts and ends at station 1, $S1 = 1$.

Three parameters "tempMax", "coefDecay" and "coefStretch" are added in the function to calculate the decreasing temperature. The main purpose is to lower the annealing speed, which allows more "jumping" from the local optimal in the early stage.

```
function [temp] = calTemp(k, tempMax, coefDecay, coefStretch)
    temp = tempMax / (1 + coefStretch * k)^coefDecay;
end
```

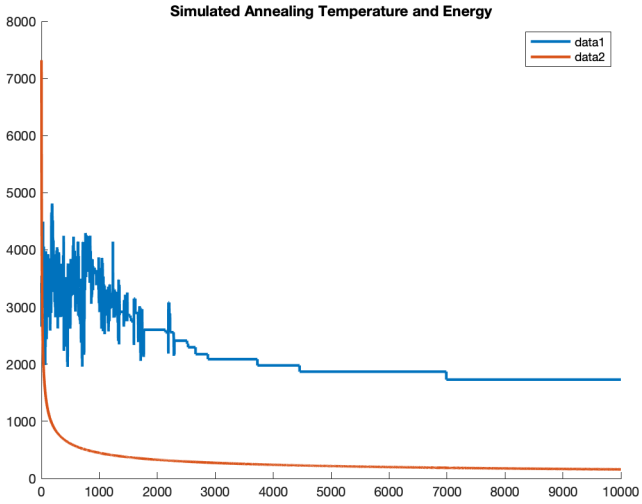

Figure 18: simulated Annealing temperature and energy.

Once there are enough "jump" for searching, we extend the simulation times to try to get better result.

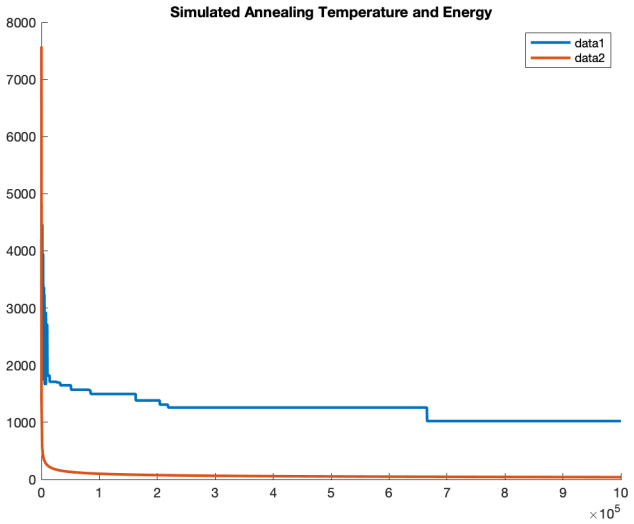

Figure 19: simulated Annealing temperature and energy.

The final result from experiment is shown in the following table:

series	startPosition	tempMax	nSample	coefDecay	coefStretch	costResult	timeElapse	ratioAccept	fig
1	1	1	100000	0.5	1	1585	2.6172	0.000140	
2	1	100	100000	0.5	1	1505	2.6622	0.005970	2
3	1	100	1000000	0.5	1	1191	27.0860	0.000571	3
4	1	100	1000000	0.4	1	1022	28.1995	0.002975	4
5	1	100	1000000	0.45	1	946	27.2704	0.001052	5

Table 8: Optimization Result.

8 Exercise 8: Bootstrap

The Bootstrap method is a technique for estimating the variance or median of an estimator based on sampling from the empirical distribution.
The aim of this exercise is using Bootstrap method to estimate variance of the estimator(according lecture 10). In this respect, at first we Simulate r data sets, each with N “observations” sampled form the empirical

distribution F_e.
For each simulated data set, estimate the parameter of interest. This is a bootstrap replicate of the estimate. Finally report the variance among the bootstrap replicates.

In this exercise we write a subroutine that takes as input a "data" vector of observed values, and which outputs the median as well as the bootstrap estimate of the variance of the median, based on r = 100 bootstrap replicates.

For testing, we Simulate $n = 200$ Pareto distributed random variates with $\beta = 1$ and $k = 1.05$.
Table 7 show the Computation of the mean, the median, and the bootstrap estimate of the variance of the sample median.

Name	Mean	lbMean	ubMean	Median	Variance	lbVar	ubVar
Simulated Pareto Dist	5.096183	29.096227	-18.903861	1.862766	172.119351	156.742619	190.867438

Name	Method	Mean	Theoretical	lbMean	ubMean	Variance	lbVar	ubVar
vecMean	bootstrap	5.182998	21	5.009419	5.356576	0.874796	1.016229	0.768077
vecMedian	bootstrap	1.884051	1.935064	1.880842	1.887261	0.016174	0.018789	0.014201

Table 9: Bootsrap Result .

From table 9, we can see that the bootstrap method can be used to estimate the sample median. The variance and confidence interval of median can also be calculated, which shows that the estimation is very accurate. For Pareto distribution, it's hard to get the precise mean. Even the variance of the estimated mean from 100 sets is large.

A Appendix: MATLAB codes

A.1 Exercise 1

```
%%
clear;
clc;
close all;
p = 10;          % classes
n = 10000;
M = 16;          % modulus
a = 5;           % multiplier
c = 1;           % shift

x=zeros(1,n);
U=zeros(1,n);

for i = 2:n
    x(:,i) = mod(a*x(:,i-1)+c,M);
    U(:,i) = x(:,i)/M;
end

rng default;
y = rand(n,1);

%%
% Histogram
figure(1)
subplot(2,1,1);
histogram(U,p);
title('Histogram of U_i');xlabel('U_i'); ylabel('Number of Occurances');

subplot(2,1,2);
histogram(y,p);
title('Histogram of y');xlabel('y'); ylabel('Number of Occurances');

% Scatter plot
figure(2)
subplot(2,1,1);
scatter(U(2:end),U(1:end-1));
title('Scatter plot of U_i'); xlabel('U_i(2:n-1)'); ylabel('U_i(1:n-1)');

subplot(2,1,2);
scatter(y(2:end),y(1:end-1));
title('Scatter plot of y'); xlabel('y(2:n-1)'); ylabel('y(1:n-1)');

% X^2
n_obs_U = hist(U,p);
n_obs_y = hist(y,p);
n_exp = n*ones(1,p)/p;
T_U = 0;
T_y = 0;

for i = 1:p
    T_U = (((n_obs_U(i)-n_exp(i))^2)/n_exp(i))+T_U;
    T_y = (((n_obs_y(i)-n_exp(i))^2)/n_exp(i))+T_y;
end

h_U = 1-chi2cdf(T_U,p-1);
h_y = 1-chi2cdf(T_y,p-1);

% Kolmogorov-Smirnov (KS)
U_s = sort(U);
U_u = unique(U_s);

ks=zeros(1,M);
for i = 1:M
```

```
    ks(i) = max(norm(U_s - U_u(i)));
end

ECDF = ks/n;
figure(3);
cdfplot(U);
hold on
cdfplot(y);
hold off
title('Kolmogorov Smirnov Test'); xlabel('U_i'); ylabel('Cumulative Probability')
legend('ECDF', 'Normal CDF')

% Run-tests
m_true = 0.5;
n1 = 0;
n2 = 0;
k = 0;
l = 0;

for i = 1:n
    if U(i) < m_true
        k = 0;
        if l == 0
            n2 = n2+1;
        end
        l = 1;
    else
        l = 0;
        if k == 0
            n1 = n1+1;
        end
        k = 1;
    end
end

N = [2*((n1*n2)/(n1+n2)),(2*n1*n2*(2*n1*n2-n1-n2))/(((n1+n2)^2)*(n1+n2-1))];

ConInt = N(1)+tinv([0.025 0.975],n-1)*sqrt(N(2));

if (ConInt(1) < n1 && n1 < ConInt(2)) && (ConInt(1) < n2 && n2 < ConInt(2))
    p = 1; % Accept
else
    p = 0; % Reject
end

% Correlation test
h=0;
for i = 1:n
    ch(i) = 1/(n-h)*U(i)*U(i+h);
end
```

A.2 Exercise 2

```
clear;
clc;
close all;
rng default
n = 10000;
p = 0.1;
U = rand(1,n);
X = floor(log(U)/log(1-p))+1;

% Plot
figure(1)
plot(sort(X));
xlabel('Outcomes'); ylabel('Geometric Random Variable');
```

```
X = [1 2 3 4 5  6];
p = [7/48 5/48 1/8 1/16 1/4 5/16];
U = rand(1,n+1);
l = length(p);

%% Direct (crude) method
CDF = zeros(1,l+1);
for  i = 2:l+1
    CDF(:,i) = CDF(:,i-1)+p(i-1);
end

for  i = 1:n
    for  j = 2:l+1
        if  U(i)<= CDF(:,j) && CDF(:,j-1)<U(i)
            OUTD(i)=X(j-1);
        end
    end
end

%% Rejection method
q=1/l;
c = max(p)/q;
ex = 100;
OUTR = [];
while length(OUTR)<n
    OUTRt = OUTR;
    for  i = 1:n
        I(i) = 1+floor(l*U(i));
        if  U(i+1) < p(I(i))/c
            OUTR(i) = I(i);
        end
    end
    OUTR = OUTR(OUTR>0);
    OUTR = [OUTR OUTRt];
    U = rand(1,n+1);
end

%% Alias method
F = l*p;
L = 1:n;
G = find(F>=1);
S = find(F<=1);
j = 1;

while ~isempty(S)
    j = S(1);
    k = G(1);
    L(j)= k;
    F(k) = F(k)-(1-F(j));
    if  F(k)<1
        G(1) = [];
        S = [S k];
    end
    S(1) = [];
    j = j+1;
end

for  i = 1:n
    I(i) = 1+floor(l*U(i));
    if  U(i+1) < F(I(i))
        OUTA(i) = I(i);
    else
        OUTA(i) = L(I(i));
    end
end
```

```
% Plot
figure(2)
subplot(3,2,1)
plot(sort(OUTD));
 title('Direct Method'); xlabel('Outcomes'); ylabel('Probability')
subplot(3,2,2);
histogram(OUTD,l);
 title('Histogram of Direct Method'); xlabel('Number of bins'); ylabel('Outcomes')

subplot(3,2,3)
plot(sort(OUTR));
 title('Rejection Method'); xlabel('Outcomes'); ylabel('Probability')
subplot(3,2,4);
histogram(OUTR,l);
 title('Histogram of Rejection Method'); xlabel('Number of bins'); ylabel('Outcomes')

subplot(3,2,5)
plot(sort(OUTA));
 title('Alias Method'); xlabel('Outcomes'); ylabel('Probability')
subplot(3,2,6);
histogram(OUTA,l);
 title('Histogram of Alias Method'); xlabel('Number of bins'); ylabel('Outcomes')

table([chi2gof(OUTD);kstest(OUTD);runstest(OUTD);corrcoef(OUTD)],...
    [chi2gof(OUTR);kstest(OUTR);runstest(OUTR);corrcoef(OUTR)],...
    [chi2gof(OUTA);kstest(OUTA);runstest(OUTA);corrcoef(OUTA)],...
    'RowNames', {'chi2 Test','Kolmogorov−Smirnov Test','Run Test',...
    'Correlation Coefficient Test'},'VariableNames', {'OUTD','OUTR','OUTA'})
```

A.3 Exercise 3, 4, 5, 6, 7, 8

See folder "code" in zip. The setup file is "setup.m". All the result is in "results" folder.