

PROBABILITY, STATISTICS AND LINEAR PROGRAMMING LAB

PAPER CODE: BS-252

Lab Manual



MAIT

उद्यमेन हि सिध्यन्ति
कार्याणि न मनोरथैः

INSTITUTE VISION

To develop a learning environment for academic excellence with technological and management competence at par with international standards and to nurture young minds with high spiritual and ethical values.

INSTITUTE MISSION

The Institute will focus to incorporate following agendas in the teaching methodology:

Engineering Hardware – Software Symbiosis

Practical exercises in all Engineering and Management disciplines will be carried out using Hardware equipment as well as the related software, enabling deeper understanding of basic concepts and encouraging inquisitive nature.

Life – Long Learning

The Institute strives to match technological advancements and encourage students to keep updating their knowledge for enhancing their skills and inculcating their habit of continuous learning.

Liberalization and Globalization

The Institute endeavors to enhance technical and management skills of students so that they are intellectually capable and competent professionals with industrial aptitude to face the challenges of globalization.

Diversification

The Engineering, Technology and Management disciplines have diverse fields of studies with different attributes. The aim is to create a synergy of the above attributes by encouraging analytical thinking.

Digitization of Learning Processes

The Institute provides seamless opportunities for innovative learning in all Engineering and Management disciplines through digitization of learning processes using analysis, synthesis, simulation, graphics, tutorials and related tools to create a platform for multi-disciplinary approach.

Entrepreneurship

The Institute strives to develop potential Engineers and Managers by enhancing their skills and research capabilities so that they become successful entrepreneurs and responsible citizens.

PROBABILITY, STATISTICS AND LINEAR PROGRAMMING LAB **PAPER CODE: BS-252**

COURSE OBJECTIVE

1. To understand probability and probability distributions.
2. To understand methods of summarization and visualization of data.
3. To understand the use of various hypothesis-testing techniques.
4. To understand methods for solving linear programming problems.

COURSE OUTCOMES

At the end of the course student will be able to:	
C.252.1	Apply probability concepts and describe probability distributions to solve real-world problems. (Unit I)
C.252.2	Analyze joint distributions and visualize data using statistical tools. (Unit II)
C.252.3	Evaluate hypothesis-testing techniques and apply regression and correlation methods for analyzing data and making predictions. (Unit III)
C.252.4	Formulate and solve linear programming problems using mathematical methods. (Unit IV)

No .	Title of Lab Experiments	CO
1.	Demonstrations of SCILAB to simple programming concepts like addition, multiplication and transpose of matrices (using loops)	CO1
2.	Fitting of Binomial distribution for given n and p	CO1
3.	Fitting of Poisson distribution after computing mean	CO1
4.	Fitting Standard Normal Distribution	CO1
5.	Compute covariance and correlation for two datasets	CO2
6.	Write a program to visualize statistical data using histogram.	CO2
7.	Fitting of regression lines	CO3
8.	Perform a Chi-square test for goodness of fit for a given dataset	CO3
9.	Perform t-test for comparing means of two samples.	CO3
10.	Program for forming first simplex table for a three variable linear programming problem.	CO4
11.	Program for solving Transportation Problem of three variables.	CO4
12.	Program for solving Assignment Problem of three variables.	CO4

EVALUATION SCHEME

	Laboratory	
Components	Internal	External
Marks	40	60
Total Marks	100	

GUIDELINES FOR CONTINUOUS ASSESSMENT FOR EACH EXPERIMENT

Attendance and performance in experiments – 40 marks

- Practical performance
- Result validation
- Attendance and Viva Questions
- Timely Submission of Lab records

The Rubrics for Experiment execution, Lab file and viva voice is given below:

Experiment Marks details:

LAC/ Sr No.	Experiment Component (LAC)	Max. Marks	Grading Rubrics	
			2 marks	1 mark
LAC 1	Practical Performance	2	Develops the program and exhibits proficiency in execution.	Not able to complete the program in the required way.
LAC 2	Result Validation	2	Accuracy of results. Demonstrates excellent understanding of the concepts relevant to the program.	Results with moderate accuracy. Demonstrates partial understanding of the concepts relevant to the program.
LAC 3	Attendance and Viva Questions	4	One mark for attendance.	
			Three marks for answering more than 75% questions.	
			Two marks for answering more than 50% questions.	
			One mark for answering less than 50% questions.	
LAC 4	Timely Submission of Lab Records	2	On time submission	Late submission

CONTENTS

Experiment 1(a): Matrix Addition	7
Experiment 1(b): Matrix Multiplication.....	9
Experiment 1(c): Matrix Transpose	11
Experiment 2: Fitting Binomial distribution for given N and P.....	13
Experiment 3: Fitting Poisson distribution By computing mean....	16
Experiment 4: Plotting Standard Normal distribution Curve.....	19
Experiment 5: Compute Covariance and Correlation for 2 datasets.....	23
Experiment 6: Visualize Statistical Data Using Histogram ...	25
Experiment 7: Fitting Regression Lines.....	27
Experiment 8: Perform a chi-square test for goodness to fit for a given data....	31
Experiment 9: Perform t-test for comparing means of two samples....	34
Experiment 10: Solving a LPP using Simplex Method	38
Experiment 11: Solving Transportation Problem	44
Experiment 12: Solving a 3 by 3 Assignment Problem.....	59

EXPERIMENT 1(A): MATRIX ADDITION

Aim/Objective: Program for finding matrix addition

Course Outcome: CO1

Software Used: SCILAB

Theory: A Matrix is a rectangular array of numbers, expressions or symbols, arranged in rows and columns. Matrix addition is commutative and associative just like addition of arithmetic numbers. Addition or subtraction can be performed on two matrices if and only if they are of same order.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \quad B = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix}$$

$$\text{Then } A \pm B = \begin{pmatrix} a_{11} \pm b_{11} & a_{12} \pm b_{12} & a_{13} \pm b_{13} \\ a_{21} \pm b_{21} & a_{22} \pm b_{22} & a_{23} \pm b_{23} \\ a_{31} \pm b_{31} & a_{32} \pm b_{32} & a_{33} \pm b_{33} \end{pmatrix}$$

Matrix Addition Algorithm:

- Declare variables and initialize necessary variables
- Enter the elements of matrices row wise using loops
- Add the corresponding elements of the matrices using nested loops
- Print the resultant matrix as console output

Sample problem: Add the matrices $A = \begin{pmatrix} 1 & \frac{5}{2} & 3 \\ \frac{5}{2} & -1 & 3 \\ 3 & 3 & 3 \end{pmatrix}$, $B = \begin{pmatrix} 0 & \frac{1}{2} & 2 \\ \frac{-1}{2} & 0 & 1 \\ -2 & -1 & 0 \end{pmatrix}$

Solution: $C = A + B = \begin{pmatrix} 1 & 3 & 5 \\ 2 & -1 & 4 \\ 1 & 2 & 3 \end{pmatrix}$

CODE:

```
m=input("enter number of rows of the Matrix: ");
n=input("enter number of columns of the Matrix: ");
disp('enter the first Matrix')
fori=1:m
forj=1:n
A(i,j)=input('\');
end
end
disp('enter the second Matrix')
fori=1:m
```

```

forj=1:n
B(i,j)=input('\');
end
end
fori=1:m
forj=1:n
C(i,j)=A(i,j)+B(i,j);
end
end
disp('The first matrix is')
disp(A)
disp('The Second matrix is')
disp(B)
disp('The sum of the two matrices is')

```

Scilab Console

```

enter number of rows of the Matrix: 2
enter number of columns of the Matrix: 2

enter the first Matrix
\1
\0
\ -2
\2

enter the second Matrix
\1
\6
\ -1
\2

The first matrix is

    1.    0.
 - 2.    2.

The Second matrix is

    1.    6.
 - 1.    2.

The sum of the two matrices is

    2.    6.
 - 3.    4.

-->|

```

```
disp(C)
```


EXPERIMENT 1(B): MATRIX MULTIPLICATION

Aim/Objective: Program to compute matrix multiplication

Course Outcome: CO1

Software Used: SCILAB

Theory: Matrix product AB is possible only if number of columns in matrix A are same as number of rows in matrix B .

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \quad B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix}$$

$$C = AB = \begin{pmatrix} c_{11} = a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & c_{12} = a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} \\ c_{21} = a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & c_{22} = a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} \end{pmatrix}$$

Note that: (i) $A_{m \times n} B_{n \times k} = C_{m \times k}$

(ii) $AB \neq BA$ in general

Matrix Multiplication Algorithm:

- Declare variables and initialize necessary variables
- Enter the elements of matrices row wise using loops
- Check the number of rows and column of first and second matrices
- If number of rows of first matrix is equal to the number of columns of second matrix, go to next step. Otherwise, print 'Matrices are not conformable for multiplication' and abort.
- Multiply the matrices using nested loops
- Print the product in matrix form as console output

Sample Problem: If $A = \begin{pmatrix} \sin x & \cos x \\ \sin x & \cos x \end{pmatrix}$ $B = \begin{pmatrix} \sin x & \sin x \\ \cos x & \cos x \end{pmatrix}$

Find AB and BA

Solution: $AB = \begin{pmatrix} \sin^2 x + \cos^2 x & \sin^2 x + \cos^2 x \\ \sin^2 x + \cos^2 x & \sin^2 x + \cos^2 x \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$

$$BA = \begin{pmatrix} 2 \sin^2 x & \sin 2x \\ \sin 2x & 2 \cos^2 x \end{pmatrix}$$

CODE:

```
m=input("Enter number of rows of the first Matrix: ");
n=input("Enter number of columns of the first Matrix: ");
p=input("Enter number of rows of the second Matrix: ");
q=input("Enter number of columns of the second Matrix: ");
```

```

ifn==p
disp('Matrices are conformable for multiplication')
else
disp('Matrices are not conformable for multiplication')
break;
end
disp('enter the first Matrix')
fori=1:m
forj=1:n
A(i,j)=input('');
end
end
disp('enter the second Matrix')
fori=1:p
forj=1:q
B(i,j)=input('');
end
end
C=zeros(m,q);
fori=1:m
forj=1:q
fork=1:n
C(i,j)=C(i,j)+A(i,k)*B(k,j);
end
end
end
disp('The first matrix is')
disp(A)
disp('The Second matrix is')
disp(B)
disp('The product of the two matrices is')
disp(C)

```

```

Scilab Console

Enter number of rows of the first Matrix: 1
Enter number of columns of the first Matrix: 2
Enter number of rows of the second Matrix: 2
Enter number of columns of the second Matrix: 1

Matrices are conformable for multiplication

enter the first Matrix
\1
\0

enter the second Matrix
\ -1
\ 2

The first matrix is

    1.    0.    2.
    2.    0.    6.

The Second matrix is

- 1.    2.
  2.    0.
  0.    0.

The product of the two matrices is

- 1.

-->

```

EXPERIMENT 1(C): MATRIX TRANSPOSE

Aim/Objective: Program to compute matrix transpose

Course Outcome: CO1

Software Used: SCILAB

Theory: The matrix A' or A^T obtained by interchanging rows and columns of a matrix A is known as its transpose.

$$A = \begin{pmatrix} 1 & 3 & 5 \\ 2 & -1 & 4 \\ 0 & 2 & 3 \end{pmatrix} \quad A^T = \begin{pmatrix} 1 & 2 & 0 \\ 3 & -1 & 4 \\ 5 & 4 & 3 \end{pmatrix}$$

Matrix Transpose Algorithm:

- Declare variables and initialize necessary variables
- Enter the elements of matrix by row wise using loop
- Interchange rows to columns using nested loops
- Print the transposed matrix as console output

CODE:

```

clc
m=input("Enter number of rows of the Matrix: ");
n=input("Enter number of columns of the Matrix: ");
disp('Enter the Matrix')
for i=1:m

```

```

for j=1:n
A(i,j)=input(' ');
end
end
B=zeros(n,m);
for i=1:n
for j=1:m
B(i,j)=A(j,i)
end
end
disp('Entered matrix is')
disp(A)
disp('Transposed matrix is')
disp(B)

```

Scilab Console

```

Enter number of rows of the Matrix: 2
Enter number of columns of the Matrix: 2

```

Enter the Matrix

```

\1
\2
\8
\6

```

Entered matrix is

```

1.  2.  2.
8.  6.  6.

```

Transposed matrix is

```

1.  8.
2.  6.

```

```

-->|

```

EXPERIMENT 2: FITTING BINOMIAL DISTRIBUTION FOR GIVEN N AND P

Aim/Objective: Program for fitting Binomial distribution when n and p are given

Course Outcome: CO1

Software Used: SCILAB

Theory: A series of independent trials which result in one of the two mutually exclusive outcomes 'success' or 'failure' such that the probability of the success (or failure) in each trials is constant, then such repeated independent trials are called as 'Bernoulli trials'. A discrete random variable which results in only one of the two possible outcomes (success or failure) is called Binomial variable.

Let there be n independent finite trials in an experiment such that

- Each trial has only two possible outcomes success and failure
- Probability of success (p) and probability of failure (q) are constant for all the trials and $p + q = 1$.

Then if a random variable X denotes the number of successes in n trials, then

$$P(X = r) = {}^nC_r p^r q^{n-r} \text{ or } P(r) = {}^nC_r q^{n-r} p^r$$

\therefore Binomial distribution may be given as $(q + p)^r$

Sample Problem: Fit a binomial distribution to the following data and compare theoretical frequencies with actual ones

x	0	1	2	3	4	5	6	7	8	9
f	6	20	28	12	8	6	0	0	0	0

Solution: Mean of the given distribution = $\frac{\sum fx}{\sum f}$, $\sum f = 80$

$$= \frac{0 + 20 + 56 + 36 + 32 + 30 + 0}{80} = \frac{87}{40} = 2.175$$

Let mean of binomial distribution to be fitted = $np = 2.175$

Also $n = 10 \therefore p = 0.2175$ $q = 1 - 0.2175 = 0.7825$

\therefore B.D. is given by $80(0.7825 + 0.2175)^{10}$

Theoretical frequencies using binomial distribution are given in the table below:

x	$P(r) = {}^nC_r q^{n-r} p^r$	Theoretical frequencies (f) = $80 \times P(r)$
0	${}^{10}C_0 (0.7825)^{10} (0.2175)^0 = 0.086$	6.9 = 7(say)
1	${}^{10}C_1 (0.7825)^9 (0.2175)^1 = 0.239$	19.1 = 19(say)
2	${}^{10}C_2 (0.7825)^8 (0.2175)^2 = 0.299$	23.9 = 24(say)
3	${}^{10}C_3 (0.7825)^7 (0.2175)^3 = 0.22$	17.8 = 18(say)
4	${}^{10}C_4 (0.7825)^6 (0.2175)^4 = 0.11$	8.6 = 9(say)
5	${}^{10}C_5 (0.7825)^5 (0.2175)^5 = 0.04$	2.9 = 3(say)

6	${}^{10}C_6(0.7825)^4(0.2175)^6 = 0.008$	$0.66 = 0(\text{say})$
7	${}^{10}C_7(0.7825)^3(0.2175)^7 = 0.001$	$0.11 = 0(\text{say})$
8	${}^{10}C_8(0.7825)^2(0.2175)^8 = 0$	0
9	${}^{10}C_9(0.7825)^1(0.2175)^9 = 0$	0

Algorithm for fitting Binomial distribution when n and p are given:

- Input n and p
- Enter the elements x_i using loop
- Enter the corresponding frequencies f_i using loop
- Calculate theoretical frequencies using binomial function
- Plot the graph between expected and theoretical frequencies using plot2d command

CODE:

```

Clearall
disp("enter no of observation")
n=input('\')
disp("value of p")
p=input('\')
disp("enter the vlaue of x")
fori=1:n
X(1,i)=input('\')
end
disp("enter no of frequency")
forj=1:n
F(1,j)=input('\')
end
EF=sum(F)*binomial(p,n-1)
disp("Given frequencies")
disp(F)
disp("Expected frequencies")
disp(EF)
plot2d3(0:n-1,F)
plot2d(0:n-1,EF)

```

```

File Edit Control Applications ?
Scilab 6.1.1 Console

"enter no of observation"
\10

"value of p"
\0.2175

"enter the vlaue of x"
\0
\1
\2
\3
\4
\5
\6
\7
\8
\9

```

```

Scilab 6.1.1 Console
File Edit Control Applications ?
Scilab 6.1.1 Console

"enter no of frequency"
\6
\20
\28
\12
\8
\6
\0
\0
\0
\0

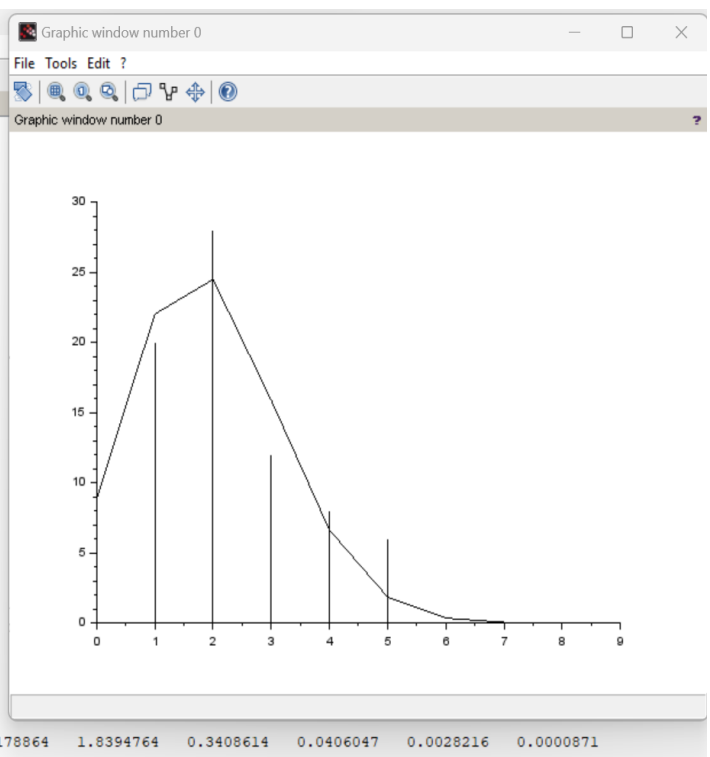
"Given frequencies"

6. 20. 28. 12. 8. 6. 0. 0. 0. 0.

"Expected frequencies"

8.7993198 22.012356 24.473802 15.872785 6.6178864 1.8394764 0.3408614 0.0406047 0.0028216 0.0000871

```



EXPERIMENT 3: FITTING POISSON DISTRIBUTION BY COMPUTING MEAN

Aim/Objective: Program for fitting Poisson distribution after computing mean

Course Outcome: CO1

Software Used: SCILAB

Theory: Poisson distribution with $P(r) = \frac{e^{-\lambda} \lambda^r}{r!}$ is a limiting case of Binomial distribution, under the conditions *i. n* $\rightarrow \infty$ *ii. p* $\rightarrow 0$ *iii. np* $= \lambda$ is finite

Sample Problem: A skilled typist kept a record of his mistakes made per day during 300 working days. Fit a Poisson distribution to compare theoretical frequencies with actual ones.

Mistakes per day	0	1	2	3	4	5	6
Number of days	143	90	42	12	9	3	1

Solution: Mean of the given distribution = $\frac{\sum fx}{\sum f}$, $\sum f = 300$

$$= \frac{0 + 90 + 84 + 36 + 36 + 15 + 6}{300} = \frac{89}{100} = 0.89 = \lambda$$

Mistakes per day	$P(r) = \frac{e^{-\lambda} \lambda^r}{r!}$	Theoretical frequency $300 \times P(r)$
0	$\frac{e^{-(0.89)} (0.89)^0}{0!} = 0.411$	123.3=123 (say)
1	$\frac{e^{-(0.89)} (0.89)^1}{1!} = 0.365$	109.5=110 (say)
2	$\frac{e^{-(0.89)} (0.89)^2}{2!} = 0.163$	48.9=49 (say)
3	$\frac{e^{-(0.89)} (0.89)^3}{3!} = 0.048$	14.4=14 (say)
4	$\frac{e^{-(0.89)} (0.89)^4}{4!} = 0.011$	3.3=3 (say)
5	$\frac{e^{-(0.89)} (0.89)^5}{5!} = 0.002$	0.6=1 (say)
6	$\frac{e^{-(0.89)} (0.89)^6}{6!} = 0.0003$	0.09=0 (say)

Algorithm for fitting Poisson distribution after computing mean:

- Input the number of observations
- Enter the elements x_i using loop
- Enter the corresponding frequencies f_i using loop
- Calculate theoretical frequencies using $P(x_i) = \frac{e^{-\lambda} \lambda^{x_i}}{r!}$
- Display $P(x_i)$

- Plot the graph between expected and theoretical frequencies using plot2d command

CODE:

```

clc
clear
disp("enter no of observation")
n=input('\n')
disp("enter the vlaue of x")
for i=1:n
    X(1,i)=input('\n')
end
disp("enter no of frequency")
for j=1:n
    F(1,j)=input('\n')
end
disp("Mean of the distribution is")
M=sum(F.*X)/sum(F)
disp(M)
for i=1:n
    P(1,i)=sum(F)*exp(-M)*M^(X(i))/factorial(X(i))
end
disp("Expected frequencies are")
disp(P)
plot2d(X,P)

```

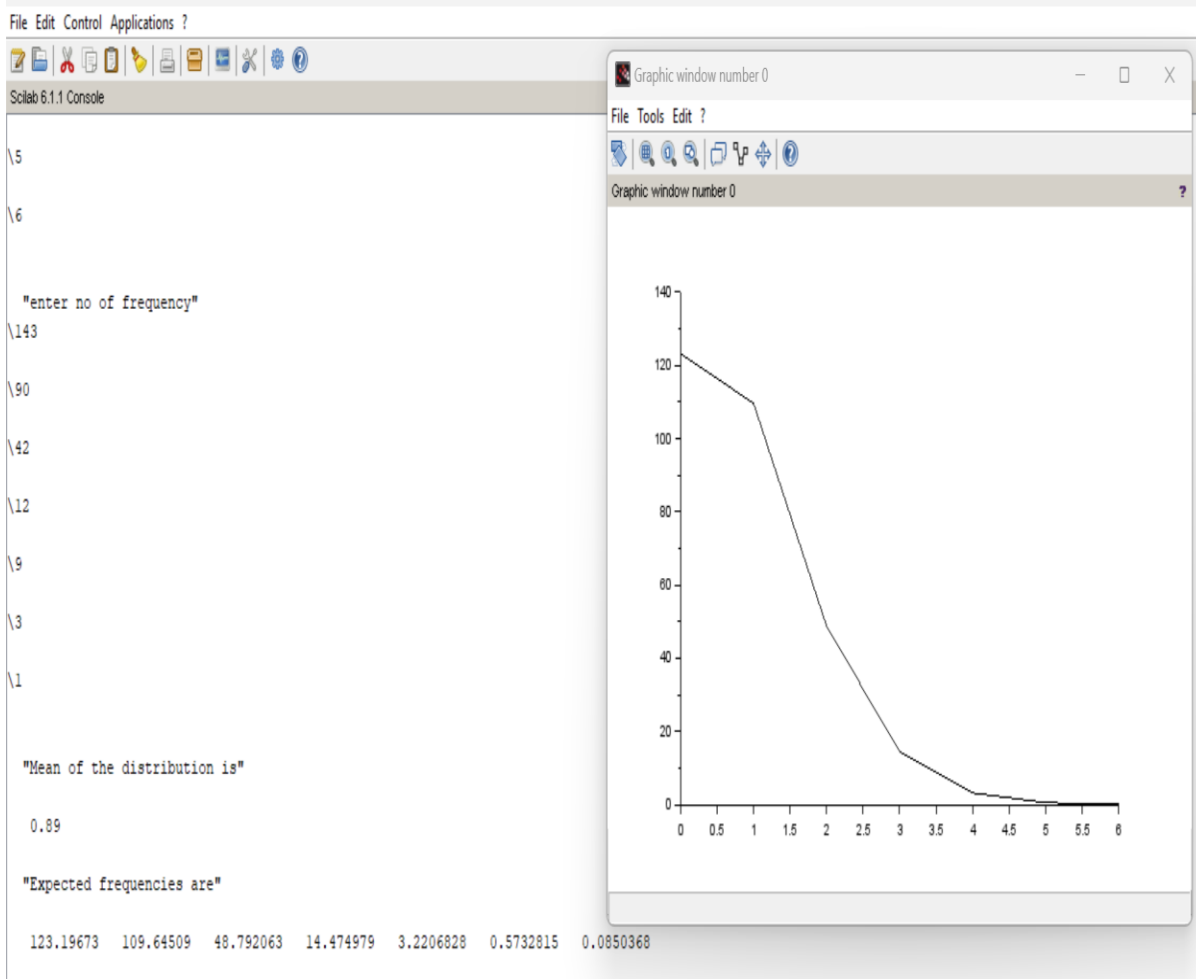
```

File Edit Control Applications ?
Scilab 6.1.1 Console

"enter no of observation"
\7

"enter the vlaue of x"
\0
\1
\2
\3
\4
\5
\6

```



EXPERIMENT 4: PLOTTING STANDARD NORMAL DISTRIBUTION CURVE

Aim/Objective: Plotting Standard Normal curve for given parameter values

Course Outcome: CO1

Software Used: SCILAB

Theory: The normal distribution developed by Gauss is a continuous distribution and is very useful in practical applications. It can be considered as the limiting form of the Binomial Distribution when the number of trials (n), is very large and neither p nor q is very small. The probability curve of a normal variate x with mean μ and standard deviation σ is given by:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, -\infty < x < \infty$$

Any normal variate x with mean μ and standard deviation σ is changed to a standard normal variate $z = \frac{x-\mu}{\sigma}$, and hence the probability density function of z is given by:

$$\phi(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2}, -\infty < z < \infty$$

The normal distribution with mean μ and variance σ^2 is denoted by $N(\mu, \sigma^2)$.

Adjoining figure shows a normal distribution curve for standard normal variate z .

Sample Problem: The daily wages of 1000 workers are normally distributed with mean 100\$ and standard deviation 5\$. Estimate the number of workers whose daily wages will be: (i) between 100\$ and 105\$ (ii) between 96\$ and 105\$

(iii) more than 110\$

(iv) more than 110\$

Solution: Let the random variable X denote the daily wages of workers in dollars.

Then X is a random variable with mean $\mu = 100$ and S.D. $\sigma = 5$.

i.e. $X \sim N(100, 25)$ and $z = \frac{X-\mu}{\sigma} = \frac{X-100}{5}$

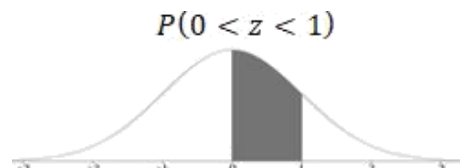
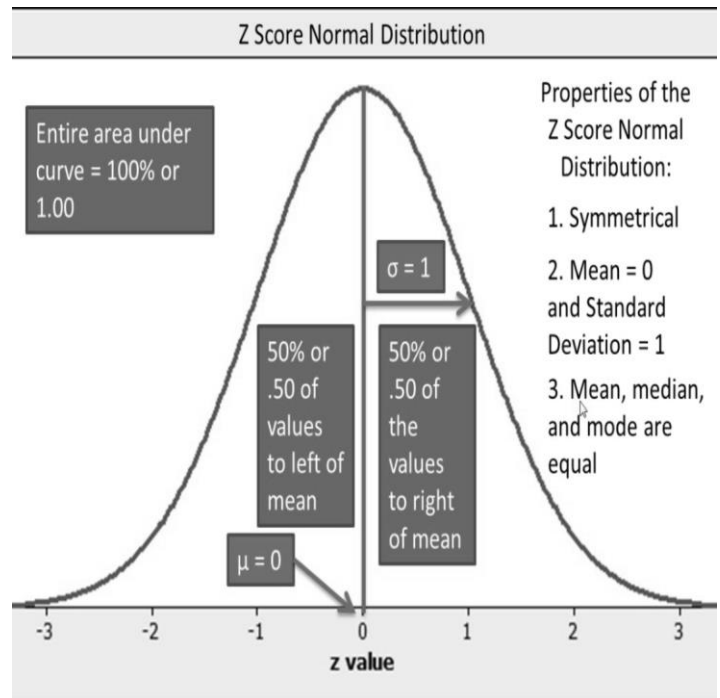
$$\Rightarrow X = 100 + 5z$$

(i) $P(100 < X < 105)$

$$= P(100 < 100 + 5z < 105)$$

$$= P\left(\frac{100-100}{5} < z < \frac{105-100}{5}\right)$$

$$= P(0 < z < 1) = 0.3413 \text{ using Z table}$$



$$(ii) P(96 < X < 105)$$

$$= P(96 < 100 + 5z < 105)$$

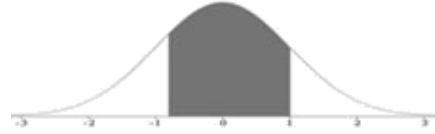
$$= P\left(\frac{96 - 100}{5} < z < \frac{105 - 100}{5}\right)$$

$$= P(-0.8 < z < 1)$$

$$= P(0 < z < 0.8) + P(0 < z < 1)$$

$$= 0.2881 + 0.3413 = 0.6294 \text{ using Z table}$$

$$P(-0.8 < z < 1)$$



$$(iii) P(X > 110)$$

$$= P(100 + 5z > 110)$$

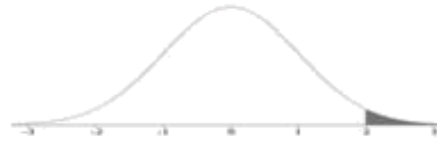
$$= P\left(z > \frac{110 - 100}{5}\right)$$

$$= P(z > 2)$$

$$= 0.5 - P(0 < z < 2)$$

$$= 0.5 - 0.4772 = 0.0228 \text{ using Z table}$$

$$P(z > 2)$$



$$(iv) P(X < 92)$$

$$= P(100 + 5z < 92)$$

$$= P\left(z < \frac{92 - 100}{5}\right)$$

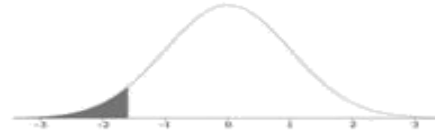
$$= P(z < -1.6)$$

$$= P(z > 1.6) \text{ (Symmetry)}$$

$$= 0.5 - P(0 < z < 1.6)$$

$$= 0.5 - 0.4452 = 0.0548 \text{ using Z table}$$

$$P(z < -1.6)$$



Algorithm:

1. Input:

- A dataset of observations (sample data).

2. Calculate the sample mean and sample standard deviation:

- The sample mean μ is the average of all data points.
- The sample standard deviation σ is the square root of the variance, which is the average of the squared differences from the mean.

3. Plot the histogram of the data:

- This helps visualize the distribution of the data.

4. Fit the normal distribution:

- Use the sample mean μ and sample standard deviation σ to fit a normal distribution.

5. Plot the fitted normal distribution curve:

- Overlay the normal distribution on the histogram of the data.

CODE:

```
// Clear workspace
clear;
clc;

// Input the dataset
x=[5.1,5.5,5.8,6.0,6.1,6.2,6.3,6.5,6.6,6.8]; // Example data

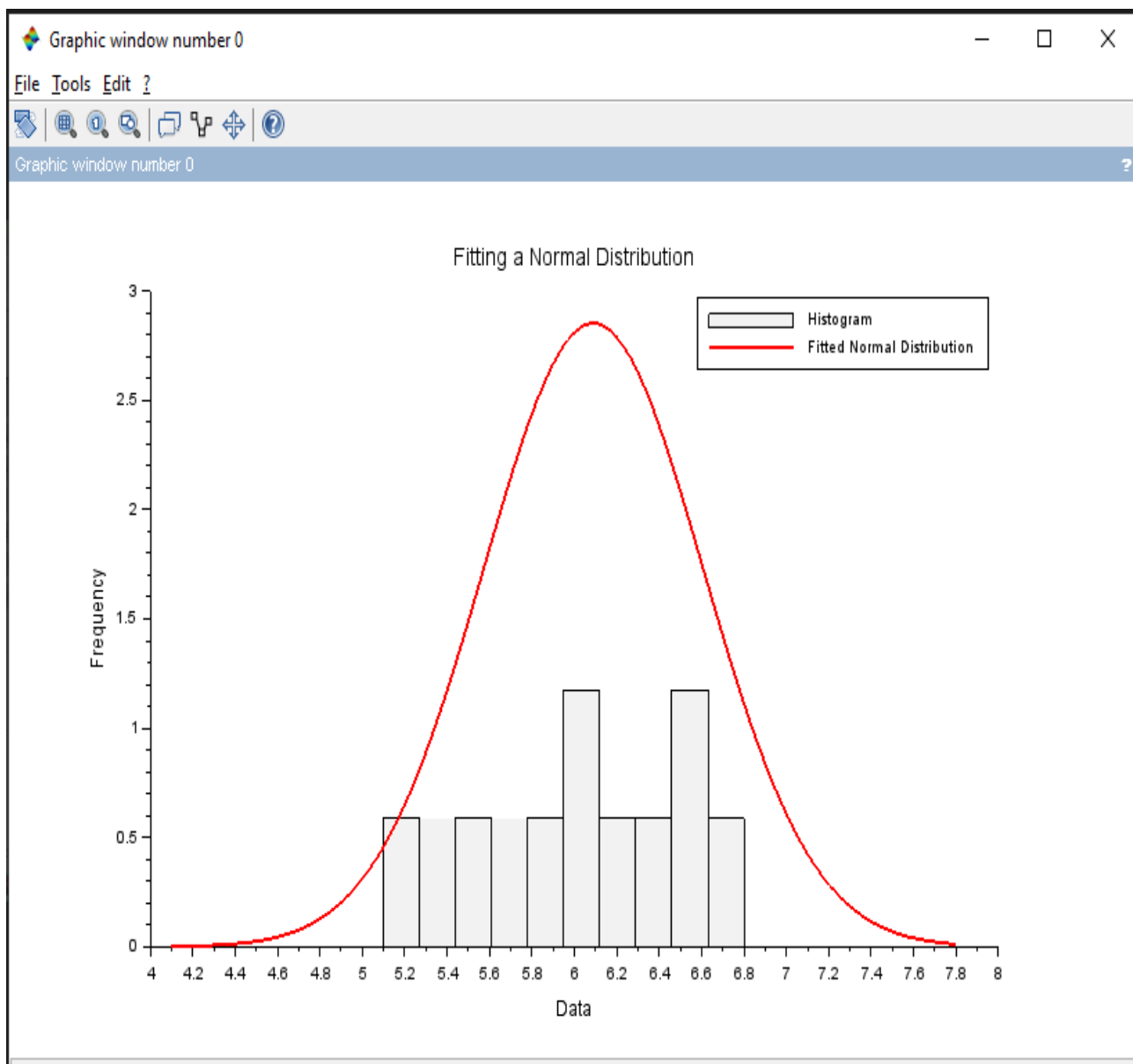
// Number of data points
n=length(x);

// Step 1: Compute mean and standard deviation
mu=mean(x);
sigma=stdev(x);

// Step 2: Define the range for the normal distribution
xmin=min(x)-1;
xmax=max(x)+1;
x_fit=linspace(xmin,xmax,100);

// Step 3: Compute the fitted normal distribution
pdf_fit=(1/(sigma*sqrt(2*pi)))*exp(-((x_fit-mu).^2)/(2*sigma^2));

// Step 4: Plot histogram and fitted normal distribution
clf; // Clear figure
histplot(10,x); // Histogram with 10 bins
plot(x_fit,pdf_fit*n*(xmax-xmin)/10,'r-','LineWidth',2); // Scale PDF to histogram
xlabel("Data");
ylabel("Frequency");
title("Fitting a Normal Distribution");
legend(["Histogram","Fitted Normal Distribution"],"location","upper_right");
```



EXPERIMENT 5: COMPUTE COVARIANCE AND CORRELATION FOR 2 DATASETS

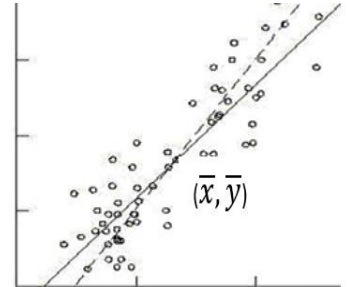
Aim/Objective: Program for computing Covariance and Correlation

Course Outcome: CO2

Software Used: SCILAB

Theory: Correlation is a measure of association between two variables; which may be dependent or independent. Whenever two variables x and y are so related; that increase in one is accompanied by an increase or decrease in the other, then the variables are said to be correlated. Coefficient of correlation (r) lies between -1 and $+1$, i.e. $-1 \leq r \leq 1$.

If r is zero; no correlation between two variables, positive correlation ($0 < r \leq +1$); when both variables increase or decrease simultaneously, and negative correlation ($-1 \leq r < 0$); when increase in one is associated with decrease in other variable and vice-versa.



4.4 Karl Pearson Coefficient of Correlation

Coefficient of correlation (r) between two variables x and y is defined as

$$r = \frac{\text{Covariance}(x,y)}{\sqrt{\text{Variance}(x)}\sqrt{\text{Variance}(y)}} = \frac{\sum d_x d_y}{\sqrt{(\sum d_x^2)(\sum d_y^2)}} = \frac{\rho}{\sigma_x \sigma_y}$$

where $d_x = x - \bar{x}$, $d_y = y - \bar{y}$, \bar{x} , \bar{y} are means of x and y data values.

$\rho = \text{Cov}(x, y) = \frac{\sum d_x d_y}{n}$ is the covariance between the variables x and y .

$$\text{Also } \sigma_x = \sqrt{\frac{\sum d_x^2}{n}} \text{ and } \sigma_y = \sqrt{\frac{\sum d_y^2}{n}}$$

Sample Problem: If $\text{Cov}(x, y) = 10$, $\text{var}(x) = 25$, $\text{var}(y) = 9$ find coefficient of correlation.

$$\text{Solution: } r = \frac{\text{Cov}(x,y)}{\sqrt{\text{var}(x)}\sqrt{\text{var}(y)}} = \frac{10}{\sqrt{25}\sqrt{9}} = \frac{10}{5 \times 3} = 0.67$$

Algorithm:

1. Input two datasets as vectors.
2. Compute the mean of each dataset.
3. Calculate the covariance using the formula:
 $\text{Cov}(X, Y) = \text{sum}((X - \text{mean}(X)) .* (Y - \text{mean}(Y))) / (n - 1)$
4. Compute the correlation using the formula:
 $\text{Corr}(X, Y) = \text{Cov}(X, Y) / (\text{std}(X) * \text{std}(Y))$
5. Display the results.

Code:

// Define the datasets

```
X=[1,2,3,4,5];// Example dataset 1
Y=[2,4,6,8,10];// Example dataset 2

// Number of data points
n=length(X);

// Compute the means
mean_X=mean(X);
mean_Y=mean(Y);

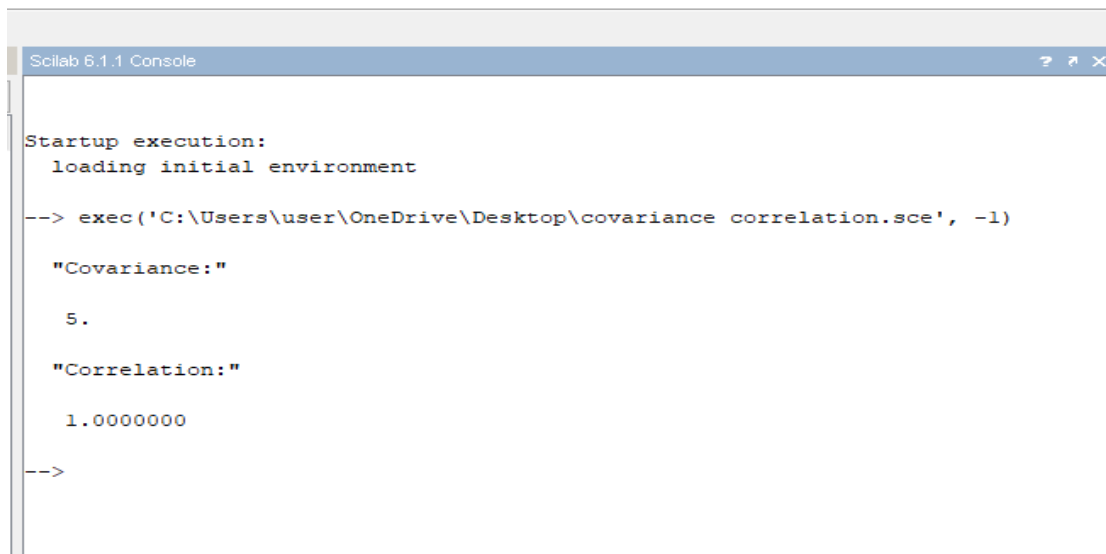
// Compute the covariance
covariance=sum((X-mean_X).*(Y-mean_Y))/(n-1);

// Compute the standard deviations
std_X=stdev(X);
std_Y=stdev(Y);

// Compute the correlation
correlation=covariance/(std_X*std_Y);

// Display the results
disp("Covariance:");
disp(covariance);
disp("Correlation:");

disp(correlation);
```



The image shows a screenshot of the Scilab 6.1.1 Console window. The window title is "Scilab 6.1.1 Console". The console output shows the startup execution, including loading the initial environment and running the command `exec('C:\Users\user\OneDrive\Desktop\covariance correlation.sce', -1)`. The output of the script is displayed as follows:

```
Startup execution:
loading initial environment

--> exec('C:\Users\user\OneDrive\Desktop\covariance correlation.sce', -1)

"Covariance:"

5.

"Correlation:"

1.0000000

-->
```


EXPERIMENT 6: VISUALIZE STATISTICAL DATA USING HISTOGRAM

Aim/Objective: Program to visualize statistical data using histogram:

Course Outcome: CO2

Software Used: SCILAB

Theory: A histogram is a visual representation of the distribution of quantitative data. It is represented by a set of rectangles, adjacent to each other, where each bar represents a kind of data.

Algorithm:

1. Input the dataset to be visualized.
2. Define the number of bins for the histogram.
3. Use the histplot function to generate and display the histogram.
4. Customize the plot with labels and title for better visualization.

Code:

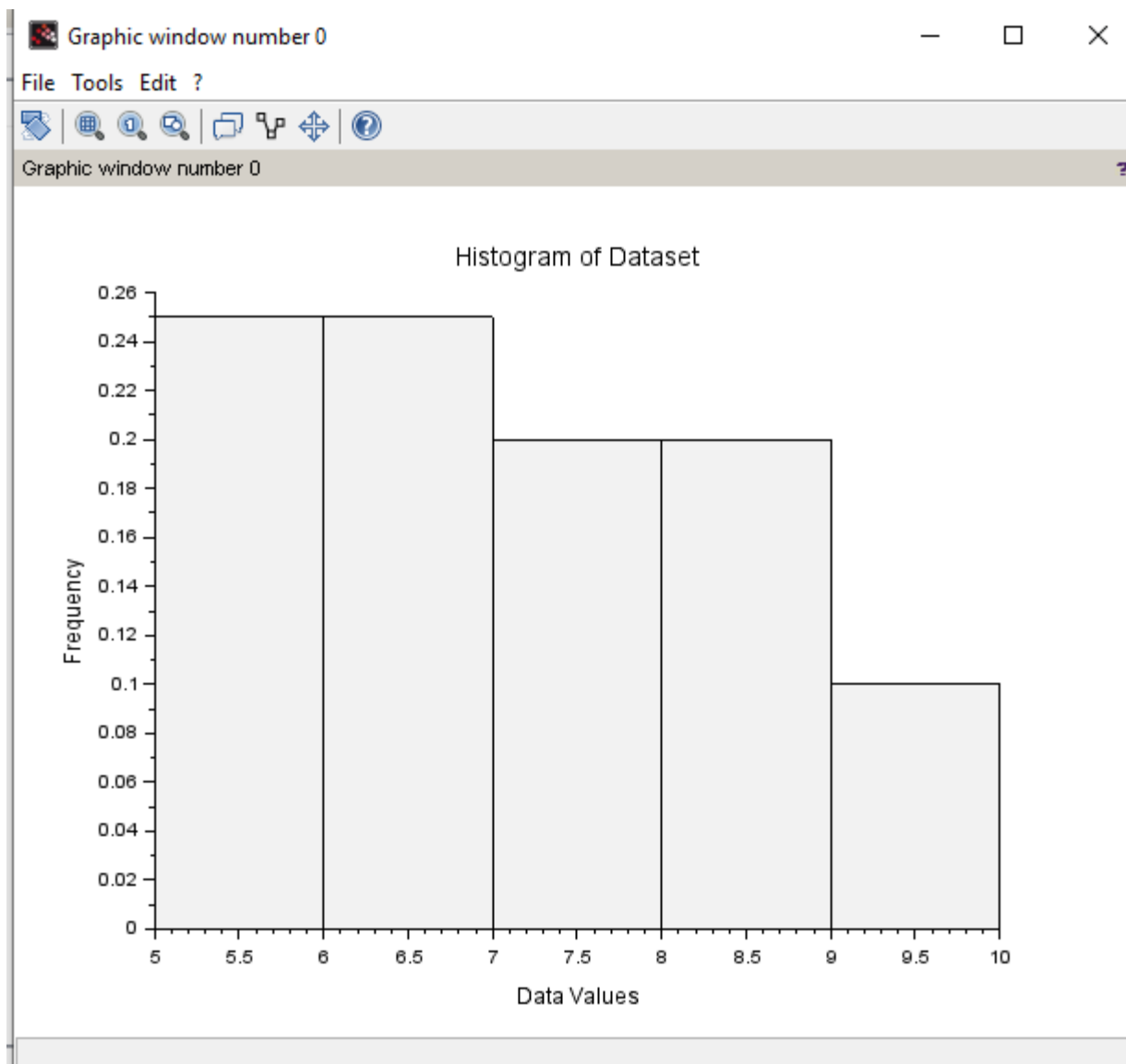
```
// Define the dataset
data=[5,7,8,9,6,7,9,8,10,7,6,8,9,5,6,7,8,9,10,7];

// Define the number of bins
num_bins=5;// Adjust this to change the bin granularity

// Generate and display the histogram
clf();// Clear any existing plots
histplot(num_bins,data);

title("Histogram of Dataset");
xlabel("Data Values");
ylabel("Frequency");

// Optionally, save the plot
// xs2png(gcf(), "histogram.png");
```



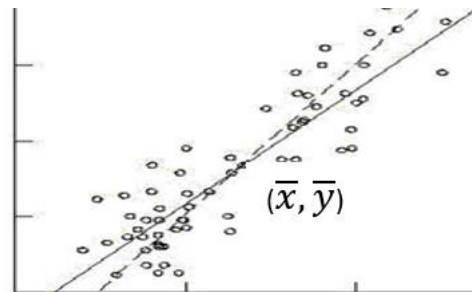
EXPERIMENT 7: FITTING REGRESSION LINES

Aim/Objective: Program for Regression lines

Course Outcome: CO3

Software Used: SCILAB

Theory: Regression describes the functional relationship between dependent and independent variables; which helps us to make estimates of one variable from the other. If we plot the observations of the linear regression between two variables, actually two straight lines can approximately be drawn through the scatter diagram. One line estimates values of y for specified values of x (known as line of regression of y on x); and other predicts values of x from given values of y (called line of regression of x on y). Here, Point of intersection of two lines of regression is (\bar{x}, \bar{y}) , Where \bar{x} and \bar{y} are the means of x series and y series.



Sample Problem: Find the regression line of y on x from the following data:

x	1	3	4	6	8	9	11	14
y	1	2	4	4	5	7	8	9

Solution: Let line of regression of y on x be:

$$y = a + bx \dots \textcircled{1}$$

Then normal equations are given by:

$$\sum y = an + b \sum x \dots \textcircled{2} \quad \text{and} \quad \sum xy = a \sum x + b \sum x^2 \dots \textcircled{3}$$

Calculating $\sum x$, $\sum y$, $\sum xy$ and $\sum x^2$

x	y	x^2	xy
1	1	1	1
3	2	9	6
4	4	16	16
6	4	36	24
8	5	64	40
9	7	81	63
11	8	121	88
14	9	196	126
$\sum x = 56$	$\sum y = 40$	$\sum x^2 = 524$	$\sum xy = 364$

Substituting values of $\sum x$, $\sum y$, $\sum xy$ and $\sum x^2$ in $\textcircled{2}$ and $\textcircled{3}$

$$\Rightarrow 40 = 8a + 56b \dots \textcircled{4}$$

and $364 = 56a + 524b \dots \textcircled{5}$

Solving $\textcircled{4}$ and $\textcircled{5}$, we get $a = \frac{6}{11}$ and $b = \frac{7}{11}$

Substituting in $\textcircled{1}$, line of regression of y on x is $y = \frac{6}{11} + \frac{7}{11}x$

Algorithm for fitting line of regression: $y = a + bx$

- Enter the number of data points
- Enter the elements x_i using loop
- Enter the corresponding points y_i using loop
- Evaluate $\sum x$, $\sum y$, $\sum xy$ and $\sum x^2$
- Solve the normal equations $\sum y = an + b \sum x$ and $\sum xy = a \sum x + b \sum x^2$ to find the values of a and b .
- The line of regression of y on x is: $y = a + bx$
- Define the x range and plot the line of regression using `plot(x, a + b * x)` command

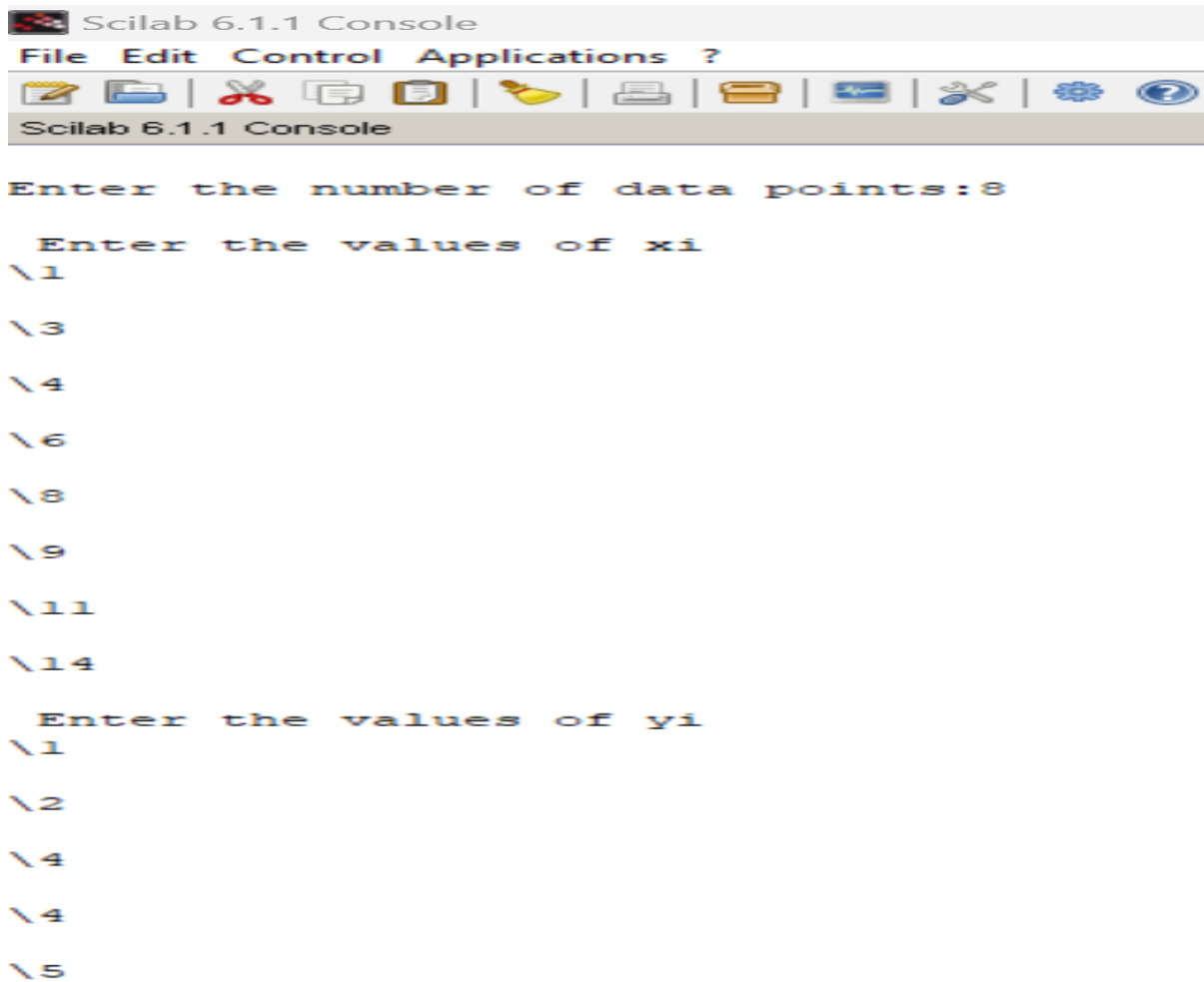
CODE:

```
clc
n=input('Enter the number of data points:')
printf(' Enter the values of xi')
for i=1:n
x(i)=input('\n');
end
printf(' Enter the values of yi')
for i=1:n
y(i)=input('\n');
end
sumx=0;sumy=0;sumxy=0;sumx2=0;
for i=1:n
sumx=sumx+x(i);
sumx2=sumx2+x(i)*x(i);
sumy=sumy+y(i);
sumxy=sumxy+x(i)*y(i);
end
a=((sumx2*sumy-sumx*sumxy)*1.0/(n*sumx2-sumx*sumx)*1.0);
b=((n*sumxy-sumx*sumy)*1.0/(n*sumx2-sumx*sumx)*1.0);
```

```
printf('The line is Y=%3.3f +%3.3f X',a,b)
```

```
x=(0:2:20)
```

```
plot(x, a+b*x)
```

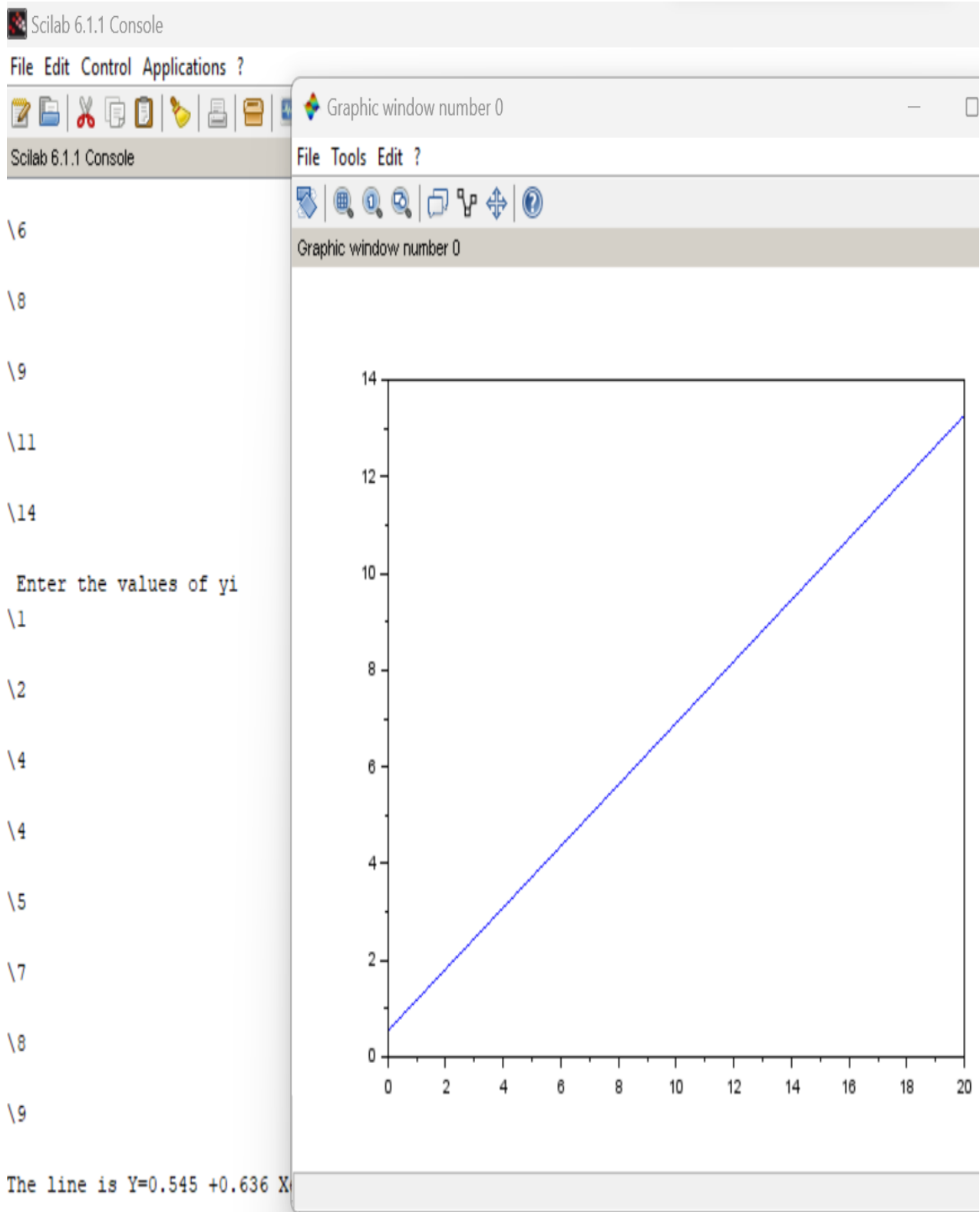


```
Scilab 6.1.1 Console
File Edit Control Applications ?
[Icons: Scissors, Copy, Paste, Eraser, Print, Save, Find, Help, etc.]
Scilab 6.1.1 Console

Enter the number of data points:8

  Enter the values of xi
\1
\3
\4
\6
\8
\9
\11
\14

  Enter the values of yi
\1
\2
\4
\4
\5
```



EXPERIMENT 8: PERFORM A CHI-SQUARE TEST FOR GOODNESS TO FIT FOR A GIVEN DATA

Aim/Objective: Program to test goodness of fit for a given dataset using Chi-square test

Course Outcome: CO3

Software Used: SCILAB

Theory: Chi-square (χ^2) is a right tailed test and describes the magnitude of discrepancy between theory and observation. If $\chi^2 = 0$; the observed and expected frequencies completely coincide, $\therefore \chi^2$ provides a measure of correspondence between theory and observations. It is one of the simplest and most general tests and can be used to perform

1. test of significance of sample variance
2. test of goodness of fit
3. test of independence of attributes in a contingency table

Chi-Square (χ^2) Test for Testing Goodness of Fit

If O_i ($i = 1, 2, \dots, n$) be a set of observed (experimental) frequencies and E_i ($i = 1, 2, \dots, n$) be the corresponding set of expected (theoretical) frequencies,

then $\chi^2 = \sum_{i=1}^n \frac{(O_i - E_i)^2}{E_i}$; with degrees of freedom (ν) = $n - 1$.

There are some underlying conditions for applying χ^2 test such as:

1. Sum of frequencies should be large (at least 50)
2. No theoretical cell-frequencies should be small, if small (less than 5) theoretical frequencies occur, regrouping of two or more cells must be done before calculating $(O_i - E_i)$. Degree of freedom is determined by the number of classes after regrouping.

Sample Problem: Apply χ^2 test of goodness to fit for the following data:

Observed frequency:	1	5	20	28	42	22	15	5	2
Theoretical Frequency:	1	6	18	25	40	25	18	6	1

Value of χ^2 for 6, 7, 8 degrees of freedom at 5% significance level are 12.592, 14.067 and 15.507 respectively.

Solution: Since first two and the last two cells frequencies are smaller than prescribed, regrouping the data to 7 cells:

O_i	6	20	28	42	22	15	7
E_i	7	18	25	40	25	18	7

Degrees of freedom (ν) = $7 - 1 = 6$

$$\begin{aligned}\chi^2 &= \sum_{i=1}^n \frac{(O_i - E_i)^2}{E_i} \\ &= \frac{(6-7)^2}{7} + \frac{(20-18)^2}{18} + \dots + \frac{(15-18)^2}{18} + \frac{(7-7)^2}{7} = 1.685\end{aligned}$$

Table value of χ^2 for 6 degrees of freedom at 5% level = 12.592

Calculated value of χ^2 is much less than table value \therefore the fit is very good.

Algorithm:

1. Input observed frequencies, expected frequencies and degrees of freedom ($d.f. = n - 1$) where n is the number of categories.
2. Compute Chi-square statistic (χ^2)
3. Compare the calculated value of χ^2 with the critical value from the chi-square table for the given degree of freedom and significance level α .
4. Decision:
 - If $\chi^2(\text{calculated}) > \chi^2(\text{critical})$, reject the null hypothesis H_0 .
 - Otherwise, fail to reject H_0 .
5. Output:
 - The χ^2 value and whether H_0 is rejected or not.

CODE:

```
// Chi-Square Test for Goodness of Fit

// Input observed and expected frequencies
observed=[50,30,20]; // Replace with your dataset
expected=[40,40,20]; // Replace with your dataset

// Check if lengths match
if size (observed,"*") <> size(expected,"*") then
error ("The observed and expected frequencies must have the same size.");
end

// Calculate chi-square statistic
chi_square_stat=sum(((observed-expected).^2)./expected);

// Degrees of freedom
n=length(observed);
df=n-1;

// Significance level
alpha=0.05;

// Critical value from chi-square distribution
chi_square_critical=cdfchi("PQ",1-alpha,df);
```

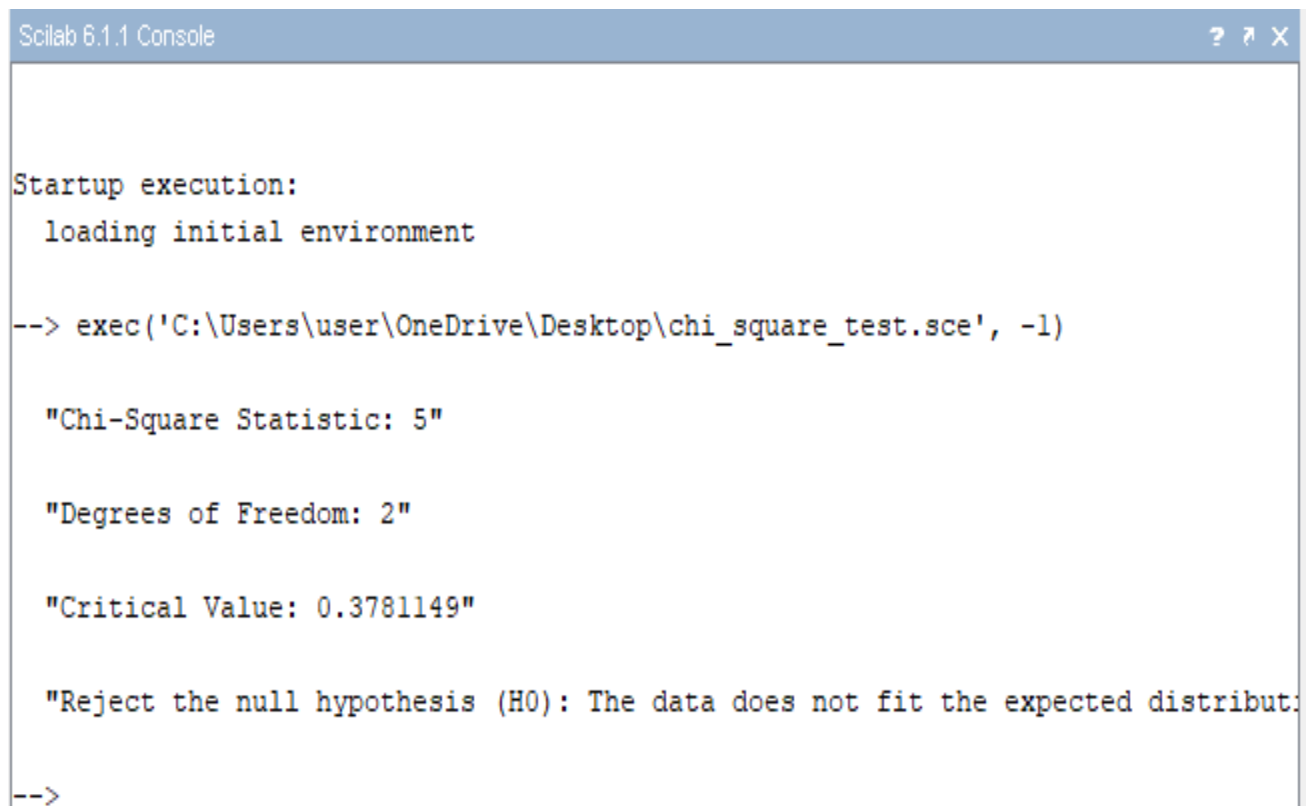


```

// Output results
disp("Chi-Square Statistic: "+string(chi_square_stat));
disp("Degrees of Freedom: "+string(df));
disp("Critical Value: "+string(chi_square_critical));

// Decision
if chi_square_stat > chi_square_critical then
disp("Reject the null hypothesis (H0): The data does not fit the expected distribution.");
else
disp("Fail to reject the null hypothesis (H0): The data fits the expected distribution.");
end

```



The image shows a screenshot of the Scilab 6.1.1 Console window. The window has a title bar with the text "Scilab 6.1.1 Console" and standard window controls (minimize, maximize, close). The console area displays the output of a script execution. It starts with "Startup execution:" followed by "loading initial environment". Then, it shows the command "--> exec('C:\Users\user\OneDrive\Desktop\chi_square_test.sce', -1)". The output of the script is displayed line by line: "Chi-Square Statistic: 5", "Degrees of Freedom: 2", "Critical Value: 0.3781149", and "Reject the null hypothesis (H0): The data does not fit the expected distribut:". The console ends with "-->".

```

Scilab 6.1.1 Console

Startup execution:
  loading initial environment

--> exec('C:\Users\user\OneDrive\Desktop\chi_square_test.sce', -1)

  "Chi-Square Statistic: 5"

  "Degrees of Freedom: 2"

  "Critical Value: 0.3781149"

  "Reject the null hypothesis (H0): The data does not fit the expected distribut:

-->

```

EXPERIMENT 9: PERFORM T-TEST FOR COMPARING MEANS OF TWO SAMPLES

Aim/Objective: Program to use t-test for comparing means of two samples

Course Outcome: CO3

Software Used: SCILAB

Theory: Gosset; who wrote under the pen-name ‘Student’, derived a theoretical distribution known as Student’s t distribution; which is used to test a hypothesis when the sample size is small and the population variance is not known. As n (size of the sample) increases; t distribution tends to normal distribution. It is evident from the graph below that t -distribution has proportionally larger area at its tails than the normal distribution.

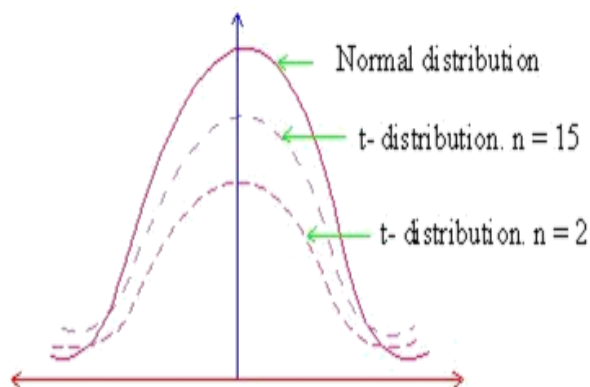
If x_1, x_2, \dots, x_n be a random sample of size $n \leq 30$, from a normal distribution with mean μ and variance σ^2 (not known); also \bar{x} be the sample mean and s standard deviation of the sample

$$\text{then } t = \frac{\bar{x} - \mu}{\frac{s}{\sqrt{n}}}, \text{ where } \bar{x} = \frac{\sum x}{n}, s = \sqrt{\frac{\sum (x - \bar{x})^2}{n-1}}$$

Degree of freedom is given by $\nu = n - 1$

Statistics t is used to test the significance of:

1. Sample mean
2. Difference between two sample means
3. Sample coefficient of correlation
4. Sample Regression Coefficient



Testing difference between means of two small samples

Let two independent samples $(x_1, x_2, \dots, x_{n_1}) ; (y_1, y_2, \dots, y_{n_2})$ of sizes $n_1 ; n_2$ with means $\bar{x} ; \bar{y}$ and standard deviations $s_x = \sqrt{\frac{\sum (x - \bar{x})^2}{n_1 - 1}} ; s_y = \sqrt{\frac{\sum (y - \bar{y})^2}{n_2 - 1}}$ be drawn from two normal populations with means $\mu_1 ; \mu_2$ and having same variance σ^2 , then to test whether the two population means μ_1 and μ_2 are same:

$$\text{Calculate } t = \frac{\bar{x} - \bar{y}}{s \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}; \text{ where } s = \sqrt{\frac{\sum (x - \bar{x})^2 + \sum (y - \bar{y})^2}{n_1 + n_2 - 2}} \text{ or } \sqrt{\frac{(n_1 - 1)s_x^2 + (n_2 - 1)s_y^2}{n_1 + n_2 - 2}}$$

and compare with table value of t (say t_1) at $(n_1 + n_2 - 2)$ degrees of freedom. Hypothesis is accepted if calculated value of $|t| < t_1$.

Sample Problem: Two independent samples showing weights in ounces of eight and seven items are given below:

Sample I:	10	12	13	11	15	9	12	14
Sample II:	9	11	10	13	9	8	10	

Is the difference between the means of the samples significant?

Value of t at 5% level of significance for 15 degrees of freedom is 2.16.

Solution: Let the null hypothesis (H_0) be: $\mu_1 \approx \mu_2$

Here $n_1 = 8$ and $n_2 = 7$, calculating sample means and standard deviations

x	$(x - \bar{x})$	$(x - \bar{x})^2$	y	$(y - \bar{y})$	$(y - \bar{y})^2$
10	-2	4	9	-1	1
12	0	0	11	1	1
13	1	1	10	0	0
11	-1	1	13	3	9
15	3	9	9	-1	1
9	-3	9	8	-2	4
12	0	0	10	0	0
14	2	4			
$\Sigma x = 96$ $\bar{x} = \frac{96}{8} = 12$		28	$\Sigma y = 70$ $\bar{y} = \frac{70}{7} = 10$		16

$$s = \sqrt{\frac{\Sigma(x-\bar{x})^2 + \Sigma(y-\bar{y})^2}{n_1+n_2-2}} = \sqrt{\frac{\Sigma(x-12)^2 + \Sigma(y-10)^2}{8+7-2}} = \sqrt{\frac{28+16}{8+7-2}} = 1.8$$

$$\therefore t = \frac{\bar{x} - \bar{y}}{s \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} = \frac{12-10}{(1.8) \sqrt{\frac{1}{8} + \frac{1}{7}}} = 2.15$$

Table value of t at 5% level of significance for 13 degrees of freedom is 2.16

\therefore Calculated value of $|t| <$ table value of t

Hence the hypothesis (H_0) is accepted that $\mu_1 \approx \mu_2$, i.e. difference between the means of the two samples is not significant.

Algorithm:

1. Input Data: Sample 1 and Sample 2.
2. Calculate Sample Means.
3. Calculate Sample Standard Deviations.
4. Calculate the Standard Error SE of the Difference in Means.
5. Calculate the T-Statistic T.
6. Determine Degrees of Freedom df.
7. Compare the T-Statistic with the Critical Value.
 - Use the t-distribution table or calculate the p-value for the given degree of freedom and significance level (e.g., $\alpha = 0.05$)

- If the absolute value of the T-statistic exceeds the critical value (or p-value is less than α), reject the null hypothesis.

CODE:

```
// T-Test for Comparing the Means of Two Samples

// Input data: Two samples
X1=[12,15,14,10,16];// Sample 1 data
X2=[22,25,20,18,21];// Sample 2 data

// Step 1: Calculate the means of both samples
mean_X1=mean(X1);
mean_X2=mean(X2);

// Step 2: Calculate the standard deviations of both samples
std_X1=stdev(X1);
std_X2=stdev(X2);

// Step 3: Calculate the standard error of the difference in means
n1=length(X1);// Sample size of X1
n2=length(X2);// Sample size of X2
SE=sqrt((std_X1^2/n1)+(std_X2^2/n2));

// Step 4: Calculate the T-statistic
T=(mean_X1-mean_X2)/SE;

// Step 5: Calculate the degrees of freedom
df=((std_X1^2/n1+std_X2^2/n2)^2)/(((std_X1^2/n1)^2)/(n1-1)+((std_X2^2/n2)^2)/(n2-1));

// Step 6: Output the results
disp("T-statistic: "+string(T));
disp("Degrees of Freedom: "+string(df));

// Step 7: Calculate the p-value for the T-statistic (two-tailed test)
p_value=2*(1-cdf("PQ",abs(T),df));// tcdf is the cumulative t-distribution function
disp("P-value: "+string(p_value));

// Step 8: Conclusion based on significance level (e.g., alpha = 0.05)
alpha=0.05;// Significance level
if p_value<alpha then
disp("Reject the null hypothesis: There is a significant difference between the means.");
else
disp("Fail to reject the null hypothesis: No significant difference between the means.");
end
```

```
Scilab 6.1.1 Console

Startup execution:
  loading initial environment

--> exec('C:\Users\user\OneDrive\Desktop\t_test.sce', -1)

  "T-statistic: -4.9331531"

  "Degrees of Freedom: 7.9587419"
cdft: Warning: using non integer values for argument #3 may lead to incorrect res

  "P-value: 0.0011623"

  "Reject the null hypothesis: There is a significant difference between the mean"

-->
```

EXPERIMENT 10: SOLVING A LPP USING SIMPLEX METHOD

Aim/Objective: Display First Simplex Table of a Linear Programming Problem

Course Outcome: CO4

Software Used: SCILAB

Theory: Simplex method is used for solving linear programming problems having two or more variables. It tests adjacent vertices of the feasible set so that at each new vertex, the value of objective function is improved until an optimal solution is attained.

Sample problem: Solve the linear programming problem:

$$\text{Max } Z = 3x_1 + 2x_2$$

Subject to

$$x_1 + x_2 \leq 4$$

$$x_1 - x_2 \leq 2$$

$$x_1, x_2 \geq 0$$

Solution: Given lpp is of maximization type and also all the b_i 's are positive.

Adding slack variables to constraints, lpp is given by:

$$\text{Max } Z = 3x_1 + 2x_2 + 0s_1 + 0s_2$$

Subject to

$$x_1 + x_2 + s_1 = 4$$

$$x_1 - x_2 + s_2 = 2$$

$$x_1, x_2, s_1, s_2 \geq 0$$

The simplex table is given as:

$C_j \rightarrow$	3	2	0	0		
B.V.	$C_B V_B$	x_1	x_2	s_1	s_2	Min Ratio $\frac{V_B}{x_k}, x_k > 0$
s_1	0 4	1	1	1	0	4
s_2	0 2	1	-1	0	1	$2 \rightarrow$
	$Z = 0$	-3 \uparrow 2	0	0		$\Delta_j = C_B x_j - C_j$
		x_1 enters and s_2 leaves				
s_1	0 2	0	2	1	-1	\rightarrow
x_1	3 2	1	-1	0	1	
	$Z = 6$	0 -5 0 \uparrow 3				$\Delta_j = C_B x_j - C_j$
		x_2 enters and s_1 leaves				

x_2	21	0	1	$\frac{1}{2}$	$-\frac{1}{2}$	
x_1	33	1	0	$\frac{1}{2}$	$\frac{1}{2}$	
	Z = 11	0	0	$\frac{5}{2}$	$\frac{1}{2}$	$\Delta_j \geq 0$

\therefore Maximum Z = 11 at $x_1 = 3$ and $x_2 = 1$

Algorithm to Display First Simplex Table of a Linear Programming Problem

- Enter the number of variables in the objective function and the number of constraints
- Using loops enter all the coefficients of the objective function and constraint equations
- Make the choices for minimization or maximization by entering 1 or 2
- Make the choices for \leq or \geq signs by entering 1 or 2
- For each \leq signs make a column for slack variable
- Print the first simplex table using all the given values

CODE:

```

clc
clear
p=input("Enter number of variables in the objective function :");
q=input("Enter number of constraint equations :");
disp('Enter coefficients of the objective function :')
for i=1:p
    A(i)=input('\');
end
m=input("Enter 1 to minimize or 2 to maximize");
disp('Enter constraint equations :')
for i=1:q
    disp('enter ' + string(i)+ ' equation: ')
    for j=1:p
        disp('enter coefficient ' + string(j)+ ' :')
        B(i,j)=input('\');
    end
    if(j==p) then
        printf("\n whether you want to maximize or minimize this equation:\nEnter \n 1. for<= \n 2. for >= :")
        C(i)=input('\');
        printf("enter constant");
        D(i)=input('\');
    end
end

```

```

        end
    end
end

printf("\n\n The LPP is      :::\n\n\n");
if(m==1) then
printf("MIN  \n");
else
printf("MAX  \n");
end
for i=1:p
    if(i==p) then
printf("%dx%d",A(i),i);
        else
printf("%dx%d + ",A(i),i);
        end
    end
printf("\nConstraint equations are \n");
for i=1:q
    for j=1:p

printf("%dx%d + ",B(i,j),j);
        if(j==p) then
            if(C(i)==1) then
printf(" <= %d",D(i));
                else if(C(i)) then
printf(" >= %d",D(i));
                    end
                end
            end
        end
    end
printf("\n");
end
printf("\n\n=====SIMPLEX TABLE
IS=====\\n\\n\n");
printf("    CJ |");
for i=1:p-1
printf(" %d",A(i));
end
for i=1:q-1

```



```

printf(" 0s%d",i);
end
printf("\n-----");
printf("\nBVCbXb |");
for i=1:p
printf(" x%d",i);
end
for i=1:q
printf(" s%d",i);
end
printf("  Min Xb/x");
printf("\n-----\n");
for i=1:p
printf("s%d  0  %d | ",i,D(i));
    for j=1:q
printf("%d  ",B(i,j));
    end
    for j=1:q
        if(j==i) then
printf("1  ");
        else
printf("0  ");
        end
    end
end
printf("\n");
end
printf("-----");
printf("\n      ZJ-CJ |");
for i=1:p
printf(" -%d",A(i));
end
for i=1:q
printf(" 0");
end

```

File Edit Control Applications ?



Scilab 6.1.1 Console

Enter number of variables in the objective function :2

Enter number of constraint equations :2

"Enter coefficients of the objective function :"

\2

\3

Enter 1 to minimize or 2 to maximize2

"Enter constraint equations :"

"enter 1 equation: "

"enter coefficient 1 :"

\2

"enter coefficient 2 :"

\5

whether you want to maximize or minimize this equation:

Enter

1. for<=

2. for >= :

\1

```

File Edit Control Applications ?
Scilab 6.1.1 Console

enter constant
\9

"enter 2 equation: "

"enter coefficient 1 : "
\1

"enter coefficient 2 : "
\3

whether you want to maximize or minimize this equation:
Enter
1. for <=
2. for >= :
\1

enter constant
\10

The LPP is      ::::

```

MAX
 $2x_1 + 3x_2$
 Constraint equations are
 $2x_1 + 5x_2 + \leq 9$
 $1x_1 + 3x_2 + \leq 10$

=====SIMPLEX TABLE IS=====

CJ 2 0s1								

BV	Cb	Xb		x1	x2	s1	s2	Min Xb/x

s1	0	9		2	5	1	0	
s2	0	10		1	3	0	1	

ZJ-CJ				-2	-3	0	0	

EXPERIMENT 11: SOLVING TRANSPORTATION PROBLEM

Aim/Objective: Solve a 3 by 3 transportation problem using MODI method with initial basic feasible solution by North West Corner Rule

Course Outcome: CO4

Software Used: SCILAB

Theory: Transportation problem is a special case of linear programming which aims to minimize the transportation cost to supply goods from various sources to different destinations, while satisfying the supply limit and demand requirement.

In general, if there be m sources and n destinations with a_i availability in i^{th} source and b_j requirement in j^{th} destination. Also ' c_{ij} ' is the cost of transportation from i^{th} source to j^{th} destination, then a transportation problem seeks to determine non-negative values of ' x_{ij} ', so as to

$$\text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n x_{ij} c_{ij}$$

$$\text{such that } \sum_{j=1}^n x_{ij} = a_i \text{ for } i = 1, \dots, m$$

$$\sum_{i=1}^m x_{ij} = b_j \text{ for } j = 1, \dots, n$$

$$x_{ij} \geq 0 \forall i, j$$

Also for a balanced transportation problem $\sum_{i=1}^m a_i = \sum_{j=1}^n b_j$

To \ From	D_1	D_2	...	D_n	Supply
S_1	c_{11} x_{11}	c_{12} x_{12}	...	c_{1n}	A_1
S_2	c_{21} x_{21}	c_{22} x_{22}	...	c_{2n}	A_2
...
S_m	c_{m1} x_{m1}	c_{m2} x_{m2}	...	c_{mn}	A_m
B_j	B_1	B_2	...	B_n	$\sum_{i=1}^m a_i = \sum_{j=1}^n b_j$

Basic feasible solution: A feasible solution of a $m \times n$ transportation problem in which allocations ' x_{ij} ' are provided satisfying the conditions $\sum_{j=1}^n x_{ij} = a_i$ and $\sum_{i=1}^m x_{ij} = b_j$ for each i and j , is said to be a basic feasible solution.

Optimal solution: A basic feasible solution which minimizes the total transportation cost is known as an optimal solution.

Sample Problem1: Obtain an initial basic feasible solution to the following transportation problem by (i) North West Corner rule (ii) Vogel Approximation method

	w_1	w_2	w_3	w_4	$a_i \downarrow$
S_1	19	30	50	10	7
S_2	70	30	40	60	9
S_3	40	8	70	20	18
$b_j \rightarrow$	5	8	7	14	

Solution: (i) Northwest Corner rule Given transportation problem is already balanced, \therefore allocating $\min [5, 7] = 5$ to cell (1,1) and adjusting supply and demand across the row and column.

	w ₁	w ₂	w ₃	w ₄	a _i ↓
S ₁	19[5]	30	50	10	7 ,2
S ₂	70×	30	40	60	9
S ₃	40×	8	70	20	18
b _j →	5	8	7	14	

Moving to right hand cell (1,2), allocating min [2,8] =2 and adjusting supply and demand across the row and column.

	w ₁	w ₂	w ₃	w ₄	a _i ↓
S ₁	19[5]	30[2]	50×	10×	7 ,2
S ₂	70×	30	40	60	9
S ₃	40×	8	70	20	18
b _j →	5	8,6	7	14	

Moving down to cell (2,2), allocating min [6,9] =6 and adjusting supply and demand across the row & column.

	w ₁	w ₂	w ₃	w ₄	a _i ↓
S ₁	19[5]	30[2]	50×	10×	7 ,2
S ₂	70×	30[6]	40	60	9 ,3
S ₃	40×	8×	70	20	18
b _j →	5	8,6	7	14	

Moving to right hand cell (2,3), allocating min [3,7] =3 and adjusting supply & demand across the row and column.

	w ₁	w ₂	w ₃	w ₄	a _i ↓
S ₁	19[5]	30[2]	50×	10×	7 ,2
S ₂	70×	30[6]	40[3]	60×	9 ,3
S ₃	40×	8×	70	20	18
b _j →	5	8,6	7,4	14	

Moving down to cell (3,3), allocating min [4,18] =4 and adjusting supply and demand across the row & column.

	w ₁	w ₂	w ₃	w ₄	a _i ↓
S ₁	19[5]	30[2]	50×	10×	7 ,2
S ₂	70×	30[6]	40[3]	60×	9 ,3
S ₃	40×	8×	70[4]	20	18 ,14
b _j →	5	8,6	7,4	14	

Allocating the balance supply/demand i.e. '14' in the cell (3,4), the initial basic feasible solution using north west corner rule is given by

	w ₁	w ₂	w ₃	w ₄	a _i ↓
S ₁	19[5]	30[2]	50×	10×	7,2
S ₂	70×	30[6]	40[3]	60×	9,3
S ₃	40×	8×	70[4]	20[14]	18,14
b _j →	5	8,6	7,4	14	

Transportation cost = 19(5) + 30(2) + 30(6) + 40(3) + 70(4) + 20(14) = 1015units

(ii) **Vogel's Approximation Method (VAM):** Writing the difference of minimum cost and next minimum cost below each column and on the right of each row. Maximum penalty is 22 against w₂, so allocating min[8,18]=8 to minimum cost cell (3,2) in w₂ column.

	w ₁	w ₂	w ₃	w ₄	a _i ↓	
S ₁	19	30×	50	10	7	(9)
S ₂	70	30×	40	60	9	(10)
S ₃	40	8[8]	70	20	18, 10	(12)
b _j →	5	8	7	14		
	(21)	(22)	(10)	(10)		

Again writing the difference of minimum cost and next minimum cost, skipping allocated cells and crossed out cells, maximum penalty is 20 against S₂ and S₃. Taking S₃ row arbitrarily and allocating min [10,14]=10 to minimum cost cell (3,4) in S₃, also adjusting corresponding supply and demand

	w ₁	w ₂	w ₃	w ₄	a _i ↓	
S ₁	19	30×	50	10	7	(9) (9)
S ₂	70	30×	40	60	9	(10) (20)
S ₃	40×	8[8]	70×	20[10]	18, 10	(12) (20)×
b _j →	5	8	7	14, 4		
	(21)	(22)	(10)	(10)		
	(21)	×(10)	(10)			

Again adjusting corresponding supply and demand

	w ₁	w ₂	w ₃	w ₄	a _i ↓	
S ₁	19[5]	30×	50	10	7, 2	(9) (9) (9)
S ₂	70×	30×	40	60	9	(10) (20) (20)
S ₃	40×	8[8]	70×	20[10]	18, 10	(12) (20) ×
b _j →	5	8	7	14, 4		
	(21)	(22)	(10)	(10)		
	(21)	×	(10)	(10)		
	(51)	(10)	(50)			

Rewriting the difference of minimum cost and next minimum cost, skipping allocated cells and crossed out cells, maximum penalty is 50 against W₄. Allocating min[2,4]=2 to minimum cost cell (1,4) in S₁, also adjusting corresponding supply and demand

	w ₁	w ₂	w ₃	w ₄	a _i ↓	
S ₁	19[5]	30×	50	10[2]	7, 2	(9) (9) (9) (40)

S ₂	70 ×	30 ×	40	60	9	(10) (20) (20) (20)
S ₃	40×	8[8]	70×	20[10]	18, 10	(12) (20) ×
b _j →	5	8	7	14, 4, 2		
(21)	(22)	(10)	(10)			
(21)	×	(10)	(10)			
(51)		(10)	(50)			
(10)	(50)	(50)				

Now allocating remaining demands 7 and 2 of W₃ and W₄, supply in S₂ is exhausted.

	w ₁	w ₂	w ₃	w ₄	a _i ↓	
S ₁	19[5]	30 ×	50×	10[2]	7, 2	(9) (9) (9) (40)
S ₂	70×	30 ×	40[7]	60[2]	9	(10) (20) (20) (20)
S ₃	40×	8[8]	70×	20[10]	18, 10	(12) (20) ×
b _j →	5	8	7	14, 4, 2		
(21)	(22)	(10)	(10)			
(21)	×	(10)	(10)			
(51)		(10)	(50)			
×	(10)	(50)				

Transportation cost = 19(5) + 10(2) + 40(7) + 60(2) + 8(8) + 20(10) = 779units

Sample Problem2: Find an optimal solution of the transportation problem given in Sample Problem1.

Solution: An
transportation

initial basic feasible solution of the
problem as given by VAM is:

	w ₁	w ₂	w ₃	w ₄
S ₁	19[5]	30	50	10[2]
S ₂	70	30	40[7]	60[2]
S ₃	40	8[8]	70	20[10]

To assign set of numbers u_i and v_j , putting $u_2 = 0$ arbitrarily as all the rows are having equal number of allocations. Calculating remaining u_i and v_j using the relation

$c_{ij} = (u_i + v_j)$ for occupied

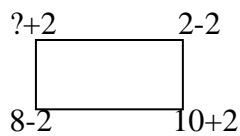
	w ₁	w ₂	w ₃	w ₄	
S ₁	19*			10 *	$u_1 = -50$
S ₂			40 *	60 *	$u_2 = 0$
S ₃		8 *		20 *	$u_3 = -40$
	$v_1 = 69$	$v_2 = 48$	$v_3 = 40$	$v_4 = 6$	

Now entering c_{ij} for each unoccupied cell (i,j) in the upper left corner and $(u_i + v_j)$ for the same cell in the upper right corner. Also writing $d_{ij} = c_{ij} - (u_i + v_j)$ at lower right corner. We observe that $d_{22} = -18$, seeking an allocation which is fulfilled by forming a closed loop with other occupied cells travelling horizontally or vertically.

w ₁	w ₂	w ₃	w ₄
----------------	----------------	----------------	----------------

S ₁	*	30	-2 +32	50	-10 +60	*	u ₁ = -50
S ₂	70	69 +1	30	? 48 -18	*	*	u ₂ = 0
S ₃	40	29 +11	*		70	0 +70	u ₃ = -40
v ₁ =69 v ₂ =48 v ₃ =40 v ₄ =60							

Adding and subtracting $\min[2,8]=2$ at vertices of the loop to adjust supply and demand requirements as given below



Modified distribution of allocations is as shown in the table below:

	w ₁	w ₂	w ₃	w ₄
S ₁	19 [5]	30	50	10 [2]
S ₂	70	30 [2]	40 [7]	60
S ₃	40	8 [6]	70	20 [12]

New transportation cost = $19(5) + 10(2) + 30(2) + 40(7) + 8(6) + 20(12) = 743$ units

To check whether the new solution is optimal, assign set of numbers u_i and v_j , putting $u_1 = 0$ arbitrarily as all the rows are having equal number of allocations. Calculating remaining u_i and v_j using the relation $c_{ij} = (u_i + v_j)$ for occupied cells.

	w ₁	w ₂	w ₃	w ₄	
S ₁	19 *			10 *	u ₁ = 0
S ₂		30 *	40 *		u ₂ = 32
S ₃		8 *		20 *	u ₃ = 10
v ₁ =19 v ₂ =-2 v ₃ =8 v ₄ =10					

Now entering c_{ij} for each unoccupied cell (i,j) in the upper left corner and $(u_i + v_j)$ for the same cell in the upper right corner.

	w ₁	w ₂	w ₃	w ₄	
S ₁	*	30 -2 +32	50 8 +42	*	u ₁ = 0
S ₂	70 41 +29	*	*	60 42 +18	u ₂ = 32
S ₃	40 29 +11	*	70 18 +52	*	u ₃ = 10
v ₁ =19 v ₂ =-2 v ₃ =8 v ₄ =10					

All $d_{ij} > 0$, \therefore the solution is optimal and minimum transportation cost = 743 units

Algorithm to solve a balanced transportation problem using MODI method with initial basic feasible solution by Northwest Corner Rule.

The Procedure is conducted in two steps:

I. Finding an initial basic feasible solution using North West Corner Rule:

1. Balance the transportation problem if not originally by adding a dummy source or destination making $\sum_{i=1}^m a_i = \sum_{j=1}^n b_j$, with zero transportation cost in added cells.
2. Start with the cell in the upper left hand corner which is north west corner (1,1) and allocate the maximum possible amount $x_{ij} = \text{Min}(a_i, b_j)$ in the cell (i, j), such that either the availability of the source S_i is exhausted or the requirement at destination D_j is satisfied or both.
3. Adjust supply and demand across the row and column in which allocation x_{ij} has been made.
4. Move to right hand cell (1,2) if there is still any available quantity left otherwise move down to cell (2,1).
5. Continue the procedure until all the available quantity is exhausted.

II. Finding the optimal solution using Modi Method

1. Find an initial basic feasible solution using Northwest Corner Rule
2. Assign a set of numbers, $u_i (i=1, 2, \dots, m)$ with each row and $v_j (j=1, 2, \dots, n)$ with each column, such that for each **occupied** cell (r,s), $c_{r,s} = u_r + v_s$. Start with a $u_i = 0$ for the row having maximum number of occupied cells and calculate remaining u_i and v_j using given relation for only **occupied** cells.
3. Enter c_{ij} for each **unoccupied** cell (i,j) in the upper left corner and enter $u_i + v_j$ for the same cell in the upper right corner.
4. Enter $d_{ij} = c_{ij} - (u_i + v_j)$ for each **unoccupied** cell at lower right corner.
5. Examine d_{ij} for unoccupied cells.
 - (i) If all $d_{ij} > 0$, solution is optimal and unique.
 - (ii) If all $d_{ij} \geq 0$, solution is optimal and an alternate optimal solution exists.
 - (iii) If least one $d_{ij} < 0$, solution is not optimal and go to step 6.
6. Form a new basic feasible solution by giving an allocation to the cell for which d_{ij} is most negative. For this **unoccupied** cell, generate a path forming a closed loop with **occupied** cells by drawing horizontal and vertical lines between them. Alternately put '+' and '-' signs at vertices of the loop starting with the **unoccupied** cell which is seeking an allocation. Add and subtract the minimum allocation where a subtraction is to be made so that one of the **occupied** cells is vacated in this process.
7. Repeat steps 2 to 5 until an optimal solution is obtained.

CODE:

```
clc
prices = [1, 2, 3;
          8, 5, 4;
          3, 1, 6]
demand = [100, 30, 70]
supply = [110, 40, 50]
prices = evstr(x_matrix('setprices', prices));
demand = evstr(x_matrix('setdemand', demand));
supply = evstr(x_matrix('submit offer', supply));
LEFT = 1
RIGHT = 2
UP = 3
DOWN = 4
// The function of calculating the cost of the transferred plan
function res=cost(prices, plan)
cntCols = length(prices(1,:))
cntRows = length(prices(:,1))
    res = 0
    for i=1:cntRows
        for j=1:cntCols
            res = res + prices(i,j) * plan(i,j)
        end
    end
endfunction
// a function that looks for available angles in a given direction
// and returns them in descending order of proximity to the edge
function [corners, success]=getAvailableCorner(basis, direction, initialPoint, i, j)
    success = 0
    corners = []
currentCorner = 1
cntCols = length(basis(1,:))
cntRows = length(basis(:,1))
colModifier = 0;
```

```

rowModifier = 0;
    if direction == LEFT then
colModifier = -1
    end
    if direction == RIGHT then
colModifier = 1
    end
    if direction == UP then
rowModifier = -1
    end
    if direction == DOWN then
rowModifier = 1
    end
i = i + rowModifier
j = j + colModifier
    while i ~= 0 && j ~= 0 && i <= cntRows && j <= cntCols
        if basis(i,j) ~= 0 || [i, j] == initialPoint then
corners(currentCorner,:) = [i,j]
currentCorner = currentCorner + 1
            success = 1
        end
i = i + rowModifier
j = j + colModifier
    end

    if success == 1 then
cornersReverse = []
        for iter = 1:length(corners(:,1))
cornersReverse(iter,:) = corners(length(corners(:,1)) - iter + 1, :)
        end
        corners = cornersReverse
    end
endfunction
// recursive looping function

```

```

function [nodes, success]=buildCycle(basis, initialPoint, currentPoint, direction)
    success = 0
    nodes = []
    possibleDirections = []
    if initialPoint == currentPoint then
    possibleDirections = [LEFT, RIGHT, UP, DOWN]
        else if direction == LEFT || direction == RIGHT then
    possibleDirections = [UP, DOWN]
        else if direction == UP || direction == DOWN then
    possibleDirections = [LEFT, RIGHT]
        end; end; end
    for directionIdx = 1:length(possibleDirections)
        [corners, suc] = getAvailableCorner(basis, possibleDirections(directionIdx), initialPoint,
currentPoint(1), currentPoint(2))
        if suc == 1 then
    possibleToCloseCycle = 0
    successWithCorners = 0
        for cornIdx = 1:length(corners(:,1))
            if (corners(cornIdx,:) == initialPoint) then
    possibleToCloseCycle = 1
                continue
            end
            [subNodes, suc] = buildCycle(basis, initialPoint, corners(cornIdx,:),
possibleDirections(directionIdx))
            if suc == 1 then
    successWithCorners = 1
            nodeIdx = 1
            nodes(nodeIdx, :) = currentPoint
                for subNodeIdx = 1:length(subNodes(:,1))
    nodeIdx = nodeIdx + 1
                    nodes(nodeIdx, :) = subNodes(subNodeIdx,:)
                end
                break
            end
        end
    end
end

```

```

        end
        if successWithCorners == 1 then
            success = 1
            break
        else if possibleToCloseCycle == 1 then
nodes(1, :) = currentPoint
nodes(2, :) = initialPoint
            success = 1
            break
        end; end
    end
end
endfunction
cntCols = length(prices(1,:))
cntRows = length(prices(:,1))
plan = [] // reference plan
plan(cntRows, cntCols) = 0 // fill with zeros
// Calculation of the original reference plan using the northwest corner method
tempDemand = demand
tempSupply = supply
for j=1:cntCols// iterate over columns (customers)
    for i=1:cntRows// iterate over rows (suppliers)
currentSupply = min(tempDemand(j), tempSupply(i))
        plan(i,j) = currentSupply
tempDemand(j) = tempDemand(j) - currentSupply
tempSupply(i) = tempSupply(i) - currentSupply

        if tempDemand(j) == 0 then
            break
        end
    end
end
end
disp("Initial plan:")
disp(plan)

```

```

printf("\nThe cost is %d \n", cost(prices, plan))
// Plan optimization
optimal = 0
UNKNOWN_POTENCIAL = 9999999
iteration = 0
while optimal ~= 1
    iteration = iteration + 1
    potencialU = []
    potencialV = []
    for i = 1:cntRows
        potencialU(i) = UNKNOWN_POTENCIAL // type unknown yet potential
    end
    for i = 1:cntCols
        potencialV(i) = UNKNOWN_POTENCIAL
    end
    potencialU(1) = 0
    continuePotentialing = 1
    // calculation of potentials by points in the route
    while continuePotentialing == 1
        continuePotentialing = 0
        // we continue to calculate the potentials if
        // for one of the values of the plan, both potentials are unknown
        for j=1:cntCols// iterate over columns (customers)
            for i=1:cntRows// iterate over rows (suppliers)
                if (plan(i,j) == 0) then
                    continue
                end
                if potencialU(i) == UNKNOWN_POTENCIAL &&potencialV(j) ==
UNKNOWN_POTENCIAL then
                    continuePotentialing = 1
                    continue
                end
                if potencialU(i) == UNKNOWN_POTENCIAL then
                    potencialU(i) = prices(i,j) - potencialV(j)

```

```

        end
        if potencialV(j) == UNKNOWN_POTENCIAL then
potencialV(j) = prices(i,j) - potencialU(i)
        end
    end
end
end
end
end
// Calculating estimates for non-basic variables
notBasis = [] // reference plan
notBasis(cntRows, cntCols) = 0 // fill with zeros
    optimal = 1
    maxI = 0;
    maxJ = 0;
    maxNB = 0;
    for j=1:cntCols // iterate over columns (customers)
        for i=1:cntRows // iterate over rows (suppliers)
            if (plan(i,j) ~= 0) then
                continue
            end
            notBasis(i,j) = potencialU(i) + potencialV(j) - prices(i,j)
            if notBasis(i,j) > 0 then
                optimal = 0
                if maxNB < notBasis(i,j) then
                    maxNB = notBasis(i,j)
                    maxI = i
                    maxJ = j
                end
            end
        end
    end
    if optimal == 1 then
        printf("Iteration %d. The current plan is optimal!", iteration)
        break
    else

```

```

printf("Iteration %d. Current plan is not optimal. Plan optimization", iteration)
end
[nodes, success] = buildCycle(plan, [maxI, maxJ], [maxI, maxJ], "")
if success == 0 then
disp("Loop building error. Shutdown")
    break
end
// Among the even nodes of the cycle (those who will have a negative 0) looking for the
minimum value
minNode = 999999999
    for node = 2:2:length(nodes(:,1))
        if minNode > plan(nodes(node, 1), nodes(node, 2)) then
minNode = plan(nodes(node, 1), nodes(node, 2))
        end
    end
    for node = 2:length(nodes(:,1))
nodeI = nodes(node, 1)
nodeJ = nodes(node, 2)
        if modulo(node, 2) == 0 then
plan(nodeI, nodeJ) = plan(nodeI, nodeJ) - minNode // for even subtract min. meaning
        else
plan(nodeI, nodeJ) = plan(nodeI, nodeJ) + minNode // for odd ones add min. meaning
        end
    end

disp("New plan:")
disp(plan)
printf("\nThe cost is %d ₺f.Đμ.\n\n", cost(prices, plan))
end
tableStr = 2;
table = []
table(1,:) = [" " "From supplier" "To the consumer" "Quantity"];
for i = 1:cntRows
    for j = 1:cntCols

```

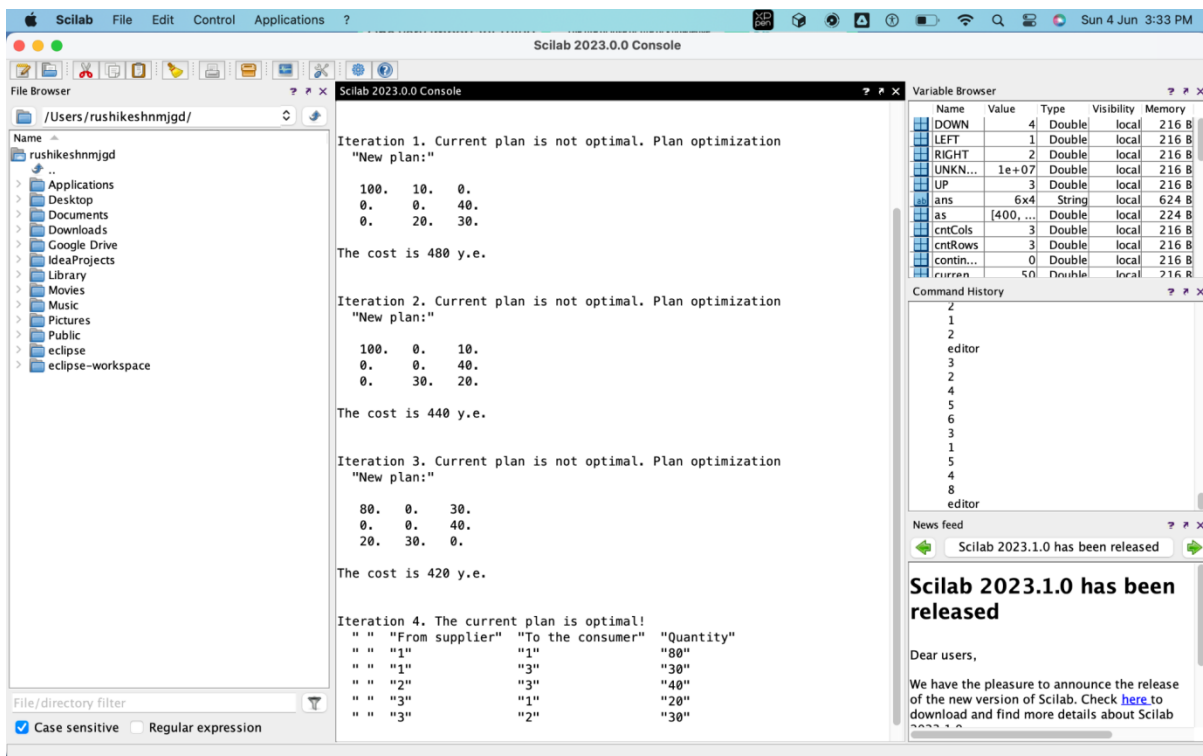
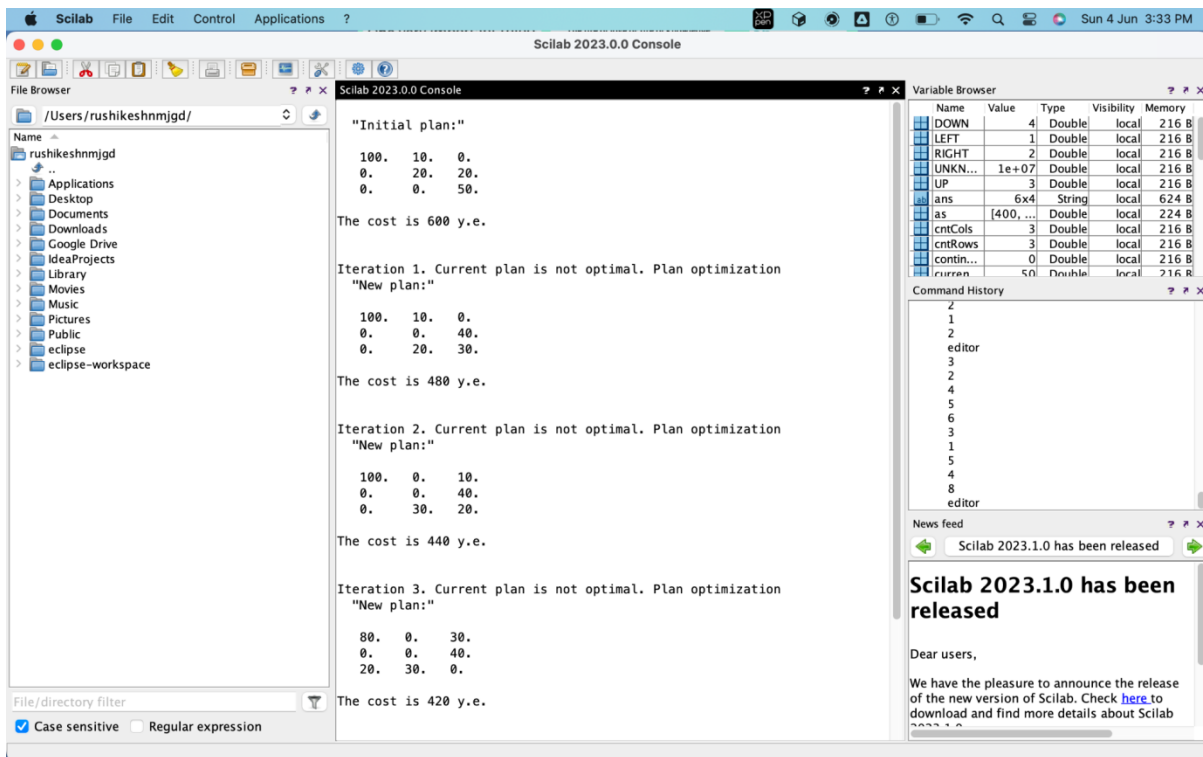


```

        if plan(i,j) ~= 0 then
str = []
str(1) = " "
str(2:4) = string([i, j, plan(i,j)])
table(tableStr,:) = str
tableStr = tableStr + 1
        end
    end
end
disp(table)
f = createWindow();
f.figure_size = [400 400];
f.figure_name = "Final answer";
as = f.axes_size;
ut = uicontrol("style", "table",...
    "string", table,...
    "position", [0 -50 400 400],...
    "tag", "Final answer");
matrix(ut.string, size(table))

```

Output:



EXPERIMENT 12: SOLVING A 3 BY 3 ASSIGNMENT PROBLEM

Aim/Objective: Solving a 3 by 3 assignment problem (Single step assignments)

Course Outcome: CO4

Software Used: SCILAB

Theory: An assignment problem is a particular case of transportation problem in which a number of operations are to be assigned to an equal number of operators, where each operator performs only one operation. The objective is to minimize overall cost or to maximize the overall profit for a given assignment schedule.

Mathematical Representation of an Assignment Problem

If there are n jobs to be performed and n persons are available for doing this job. Assume all persons can do each job at a time with different payoffs. Let c_{ij} be the cost associated with i^{th} person assigned with j^{th} job. Then the problem is to find the values ' x_{ij} ' ($x_{ij} = 0$ or 1), so that total cost for performing all the jobs is minimum.

$$\text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

Such that

$$\sum_{j=1}^n x_{ij} = 1 \text{ for } i = 1, \dots, n$$

$$\sum_{i=1}^n x_{ij} = 1 \text{ for } j = 1, \dots, n$$

$$x_{ij} = 1 \text{ if } i^{\text{th}} \text{ person is assigned with } j^{\text{th}} \text{ job, } 0 \text{ otherwise}$$

Algorithm for solving a 3 by 3 assignment problem (Single step assignments)

1. Prepare a square 3×3 matrix. If not, make it square by adding suitable number of dummy rows (or columns) with zero cost elements.
2. Subtract the minimum element of each row from every element of that row.
3. Further modify the resulting matrix by subtracting the minimum element of each column from all the elements of the respective columns, so that each row and column is having a zero element.
4. Now start making assignments row - wise. Examine each row one by one and assign a ' \square ' to '0' in the rows having single zeros. Then, mark a ' \times ' to all zeroes in the column in which assignment has been made so that no other assignment can be made in the same column.
5. Repeat the procedure for the columns.
6. Stop if all 3 assignments have been made.

Sample problem: A department head has four subordinates and four tasks have to be performed. Time (hours) each man would take to perform each task is given below. How the tasks should be allocated to each subordinate so as to minimize the total man-hours?

		Subordinates			
		S ₁	S ₂	S ₃	S ₄
Tasks	T ₁	5	6	8	9
	T ₂	6	8	10	6
	T ₃	9	5	8	5
	T ₄	9	8	7	10

Solution: Given assignment problem is balanced i.e. number of rows and columns are equal. Subtracting minimum element in each row from all the elements of the row, hence inducing a zero element in each row

	S ₁	S ₂	S ₃	S ₄
T ₁	0	1	3	4
T ₂	0	2	4	0
T ₃	4	0	3	0
T ₄	2	1	0	3

Each column is having a zero element, so matrix is unchanged by subtracting minimum element in each column. Now making assignments on zeros to rows having single zeros and crossing out remaining zeros in same columns and repeating the process with columns if any allocations are left in row assignments

	S ₁	S ₂	S ₃	S ₄
T ₁	0	1	3	4
T ₂	0	2	4	0
T ₃	4	0	3	0
T ₄	2	1	0	3

All the assignments have been made in single step.

Optimal assignments: T₁ - S₁ , T₂ - S₄ , T₃ - S₂ , T₄ - S₃

Minimum man hours = (5+6+5+7) hours = 23 hours

CODE:

```
clc
clear
n=input("how many workers and job do you have      :");
printf("\nenter time of\n\n");
for i=1:n
    for j=1:n
        printf("worker %d  job%d :",i,j);
        T(i,j)=input('\');
    end
end
printf("\nDATA YOU ENTERED IS :\n ");
for i=1:n
    printf("job%d ",i);
end
for i=1:n
    printf("\nworker%d ",i);
    for j=1:n
        printf("%d ",T(i,j));
    end
end
minim=[1000,1000,1000,1000,1000,1000,1000,1000,1000];
for i=1:n
    for j=1:n
        if(T(i,j)<=minim(i)) then
            minim(i)=T(i,j);
        end
    end
end
printf("\n");
for i=1:n
    for j=1:n
        T(i,j)=T(i,j)-minim(i);
    end
end
```

```

end
printf("\n");
printf("\n\n*****data after row minimum decrement is*****\n\t");
for i=1:n
    printf("job%d ",i);
end
for i=1:n
    printf("\nworker%d ",i);
    for j=1:n
        printf("%d ",T(i,j));
    end
end
zerr=[1000,1000,1000,1000,1000,1000];
zerc=[1000,1000,1000,1000,1000,1000];
for i=1:n
    for j=1:n
        if(T(i,j)==0) then
            zerr(i)=0;
            zerc(j)=0;
        end
    end
end
f=0;
y=0;
for i=1:n
    if(zerr(i)) then
        f=1;
    end
end
for i=1:n
    if(zerc(i)) then
        y=1;
    end
end
if((f==1)||(y==1)) then

```

```

mn=[1000,1000,1000,1000,1000,1000,1000,1000,1000];

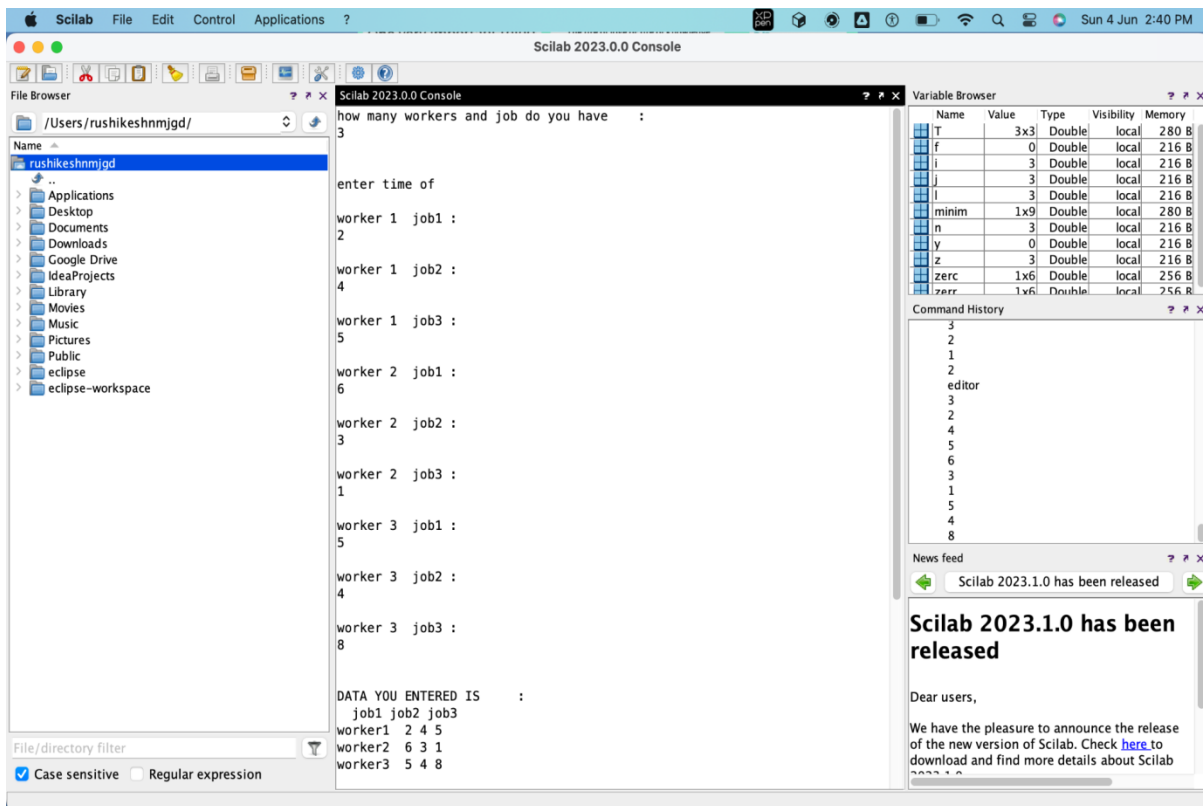
    for j=1:n
        for i=1:n
            if(T(i,j)<=mn(j)) then
mn(j)=T(i,j);
                end
            end
        end
    for j=1:n
        for i=1:n
            T(i,j)=T(i,j)-mn(j);
        end
    end
printf("\n\n*****data after column minimum decrement is*****\n");
for i=1:n
    printf("job%d ",i);
end
for i=1:n
    printf("\nworker%d ",i);
    for j=1:n
        printf("%d ",T(i,j));
    end
end
end
printf("\n\n*****Final job assignment is*****");
for i=1:n
    for j=1:n
        if(T(i,j)==0) then
printf("\n assign job %d to worker %d",j,i);
            for z=1:n
                if (z==i) then
                    continue;
                end
                if(T(z,j)==0) then

```

```

        T(z,j)=1000;
    end
    for l=1:n
        if(l==j) then
            continue;
        end
        if(T(i,l)==0) then
            T(i,l)=1000;
        end
    end
end
end
end
end
end
printf("\n");

```



Scilab 2023.0.0 Console

File Browser: /Users/rushikeshnmjgd/

Scilab 2023.0.0 Console

```

worker 2 job1 :
6

worker 2 job2 :
3

worker 2 job3 :
1

worker 3 job1 :
5

worker 3 job2 :
4

worker 3 job3 :
8

DATA YOU ENTERED IS :
  job1 job2 job3
worker1 2 4 5
worker2 6 3 1
worker3 5 4 8

*****data after row minimum decrement is*****
      job1 job2 job3
worker1 0 2 3
worker2 5 2 0
worker3 1 0 4

*****Final job assignment is*****
assign job 1 to worker 1
assign job 3 to worker 2
assign job 2 to worker 3

```

Variable Browser

Name	Value	Type	Visibility	Memory
T	3x3	Double	local	280 B
f	0	Double	local	216 B
i	3	Double	local	216 B
j	3	Double	local	216 B
l	3	Double	local	216 B
minim	1x9	Double	local	280 B
n	3	Double	local	216 B
y	0	Double	local	216 B
z	3	Double	local	216 B
zerc	1x6	Double	local	256 B
zerr	1x6	Double	local	256 B

Command History

```

3
2
1
2
editor
3
2
4
5
6
3
1
5
4
8

```

News feed

Scilab 2023.1.0 has been released

Scilab 2023.1.0 has been released

Dear users,

We have the pleasure to announce the release of the new version of Scilab. Check [here](#) to download and find more details about Scilab 2023.1.0.