

DEPARTMENT OF MATHEMATICS
(Under DBT Star College Scheme)

LABORATORY MANUAL



Programming in SCILAB

24CMAT2P



G.T.N. ARTS COLLEGE (AUTONOMOUS)

A Telugu Linguistic Minority Institution | Affiliated to Madurai Kamaraj University
Re-accredited (Cycle - II) by NAAC with 'A++' Grade
Recognized under DBT STAR COLLEGE SCHEME | Supported by DST - FIST Program
Listed in NIRF- 2024 Ranking
Old Karur Road, Dindigul – 624 005.

Sl. No.	Program	Page No.
	Introduction	1
	Syntax	8
1.	Find the cardinality of the sets.	10
2.	Discuss inclusion and exclusion problem.	11
3.	Evaluate the Polynomial.	12
4.	Compute first order and second order differentiation of functions.	13
5.	Compute single integral, double integral and triple integral of functions.	14
6.	Compute permutations and combinations.	15
7.	Discuss sum rule principle and product rule principle.	17
8.	Compute conditional probability.	18
9.	Plot 2D and 3D graphs.	19
10.	Compute Matrix Addition and Multiplication	21

Introduction to SCILAB

Scilab is short used for science laboratory. Scilab was created in 1990 by researchers from INRIA and École nationale des ponts et chaussées (ENPC). Scilab is free and open-source software for numerical computation providing a powerful computing environment for engineering and scientific applications.

Scilab is released as open source under the GPL License, and is available for download free of charge. Scilab is available under GNU/Linux, Mac OS X and Windows XP/Vista/7/8/10 (see system requirements).

What does Scilab do?

Scilab includes hundreds of mathematical functions. It has a high-level programming language allowing access to advanced data structures, 2-D and 3-D graphical functions.

A large number of functionalities is included in Scilab such as

- **Maths & Simulation:** For usual engineering and science applications including mathematical operations and data analysis.
- **2-D&3-D Visualization:** Graphics functions to visualize, annotate and export data and many ways to create and customize various types of plots and charts.
- **Optimization:** Algorithms to solve constrained and unconstrained continuous and discrete optimization problems.
- **Statistics:** Tools to perform data analysis and modeling
- **Control Systems:** Standard algorithms and tools for control system study
- **Signal Processing:** Visualize, analyze and filter signals in time and frequency domains.
- **Application Development:** Increase Scilab native functionalities and manage data exchanges with external tools.
- **Xcos - Dynamic systems modeling:** Modeling mechanical systems, hydraulic circuits, control systems...

Scilab is based on methods of numerical computation:

- Data analysis
- Algorithm development
- Model

Available methods:

- Linear algebra, sparse matrices
- Polynomials and rational functions
- Interpolation and regression
- Differentiation and integration
- Linear systems of equations
- Fourier analysis
- Ordinary Differential Equation solver and Differential Algebraic Equations solver
- Statistics

Released on Thu, 24 Oct 2024

[System requirements](#) | [Change log](#)

Scilab 2025.0.0 is released under the terms of the [GNU General Public License \(GPL\) v2.0](#).

Windows 8, 10, 11

[Scilab 2025.0.0 - Windows 64 bits \(exe\)](#)

This version has been compiled by Dassault Systèmes and other builds are available on [Gitlab](#).

GNU/Linux

[Scilab 2025.0.0 - Linux 64 bits \(tar.xz\)](#)

This version has been compiled by Dassault Systèmes and other builds are available on [Gitlab](#).

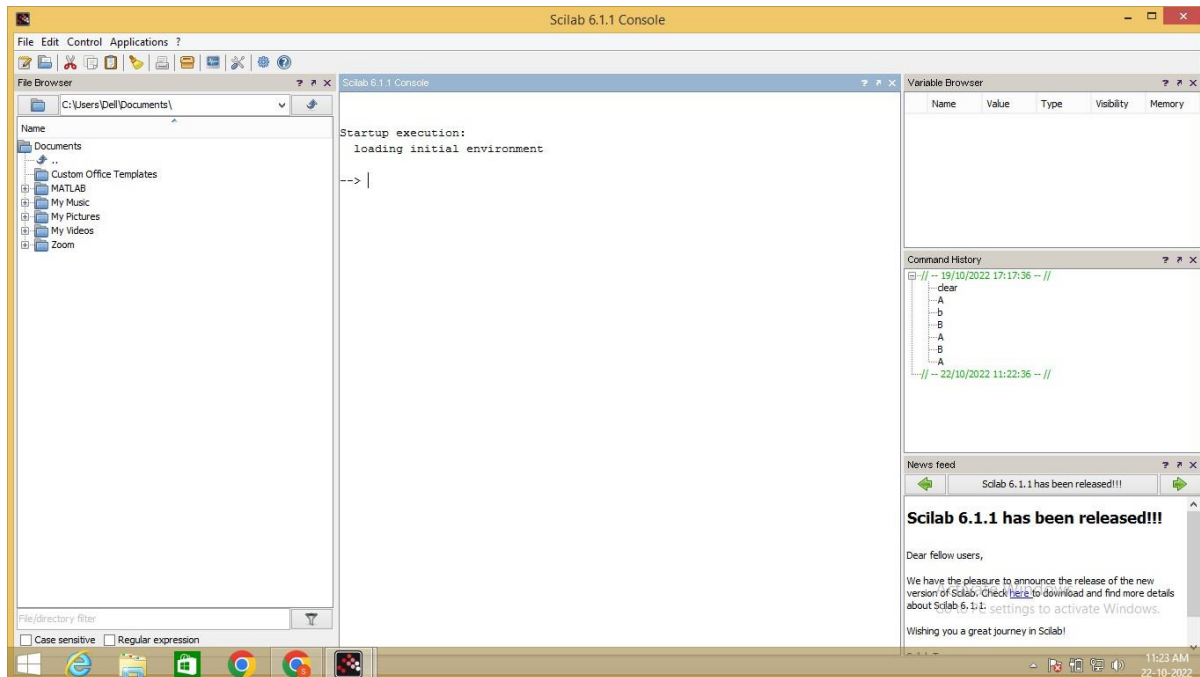
macOS

[Scilab 2025.0.0 - macOS 64 bits \(Intel\) \(dmg\)](#)

[Scilab 2025.0.0 - macOS 64 bits \(ARM\) \(dmg\)](#)

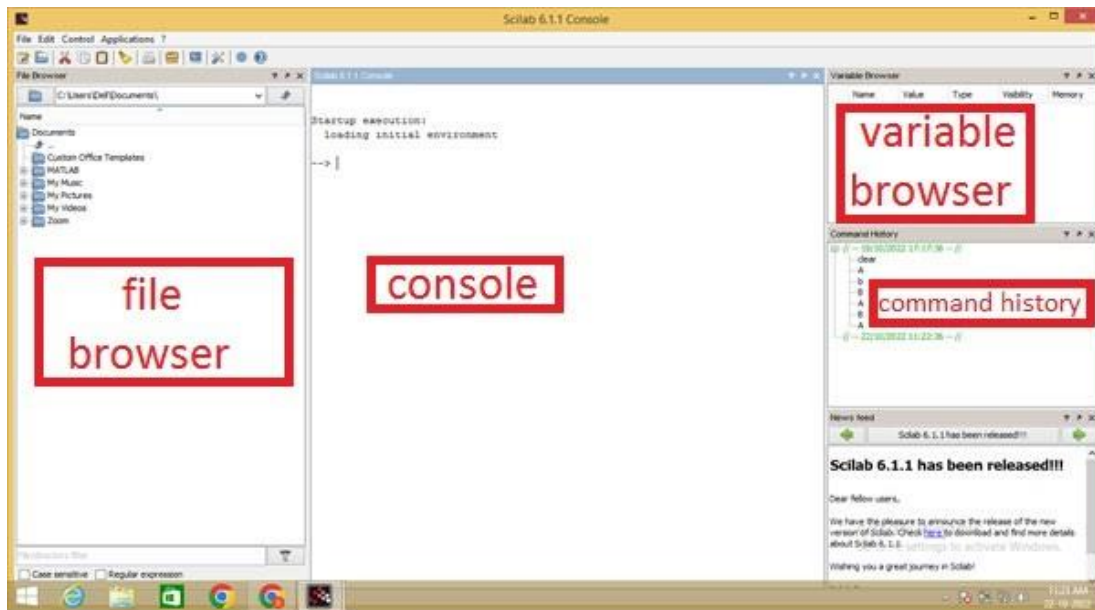
Getting started with Scilab:

Installation of the software is easy like any other software. The software gets ready after its installation into a computer system. Just like any other software, Scilab can be opened by Double clicking on the Scilab icon in desktop or by clicking on the icon appearing after the entering the name Scilab in the Windows search bar.



This window is the default layout of the Scilab desktop. It is a set of tools for managing files, variables, and applications associated with Scilab.

1. The Console is a command window used for entering Scilab functions and other commands at the command line prompt appearing as -->
2. The Command History Window is used to view or execute previously run functions.
3. The Current Directory/Workspace Window lists the folders/files in the Current Directory (where you are working) or the values and attributes of the variables you have defined.
4. The variable browser allows you to find the variables previously used during the current session.



Practicing Scilab environment with simple commands to familiarize Command Window, History, Workspace, Current Directory, Figure window, Edit window, Shortcuts, Help files.

SOURCE CODE:

help – detailed help menu for scilab commands
 who – list all the variables from the variable browser window
 whos - list all the variables with byte size, variable type etc.
 clc – clears command window
 clear – removes the variable from memory
 quit – to close the session

Pre-defined variables

%pi – 3.14
 % e = 2.718 (Euler's constant)
 %eps – epsilon
 %inf – infinity
 %i – the imaginary number i

Scilab being a matrix environment, treats all entries as a matrix. There are many built in functions to create different matrices that is listed below

Using Scilab as a calculator

The basic arithmetic operators are + for addition, - for subtraction, * for multiplication, / for division, ^ for exponentiation, and these are used in conjunction with braces or commonly called round brackets (). The symbol ^ is used to get exponents (powers): $2^4 = 16$. An alternative symbol used for the same purpose is **.

Example: --> $2 + 3/4 * 5$

```
--> 2+3/4*5
ans =

5.75
```

Note that in this calculation the result was $2+(3/4)*5$ and not $2+3/(4*5)$ because Scilab works according to the priorities of operations given in the following order.

1. quantities in brackets ()
2. powers or exponent ^ or **
3. multiplication *, left division /, and right division \, working left to right
4. addition + and subtraction -, working left to right

Assignment Operator

The assignment operator “=” is used for assigning a value (or a matrix) to variable. On the left-hand side of “=” is placed a variable name to which the value on right hand side is to be assigned. For example, let us observe the following Scilab command demonstrating the use of assignment operator.

Variable Names

Variable names may be as long as the user wants, but only the first 24 characters are taken into account in Scilab. For consistency, we should consider only variable names which are not made of moreun than 24 characters. All ASCII letters from "a" to "z", from "A" to "Z" and from "0" to "9" are allowed with the additional letters "%", "_", "#", "!", "\$", "?".

Basic Scilab Commands for Mathematical Computations

Scilab is a powerful open-source software for numerical computation. Below are some fundamental commands used in Scilab for mathematical operations:

Displaying Output in the Console:

Automatic Display: Scilab usually displays the result of any command you type and execute. For example, typing $2 + 2$ and pressing Enter will show `ans = 4`. `ans` stores the last unassigned result.

disp() function: `disp()` displays variables or strings. It's useful for formatted output or messages. For example, `x = 10; disp(x);` displays 10. `disp("The value of x is: " + string(x));` displays "The value of x is: 10". `disp("Hello, World!");` displays "Hello, World!".

Method to take input in the Console:

Assigning values: `x = 10; y = "hello"; A = [1 2 3; 4 5 6];`

Using input(): `name = input("Enter your name: ", "string"); age = input("Enter your age: ");`

1. Basic Arithmetic Operations

- Addition: `a + b`
- Subtraction: `a - b`
- Multiplication: `a * b`
- Division: `a / b`
- Power: `a ^ b`

2. Mathematical Functions

- Square root: `sqrt(x)`
- Exponential: `exp(x)`
- Logarithm (natural): `log(x)`
- Logarithm (base 10): `log10(x)`
- Sine: `sin(x)`
- Cosine: `cos(x)`
- Tangent: `tan(x)`
- Absolute value: `abs(x)`

3. Matrix Operations

- Creating a matrix: `A = [1 2 3; 4 5 6; 7 8 9]`
- Transpose: `A'`

- Inverse: `inv(A)`
- Determinant: `det(A)`
- Identity matrix: `eye(n)` (creates an $n \times n$ identity matrix)
- Zeros matrix: `zeros(m, n)` (creates an $m \times n$ matrix of zeros)
- Ones matrix: `ones(m, n)` (creates an $m \times n$ matrix of ones)

4. Solving Equations

- Solving a system of equations: $x = A^{-1} B$ (for $Ax = B$)
- Polynomial roots: `poly([2 -3 1])` (finds roots of $x^2 - 3x + 2 = 0$)

5. Defining Variables & Functions

Assigning a value: $x = 5$

Defining a function:

```
function y = f(x)
    y = x^2 + 2*x + 1;
endfunction
```

6. Loops and Conditional Statements

For Loop:

```
for i = 1:10
    disp(i)
end
```

While Loop:

```
i = 1;
while i <= 10
    disp(i)
    i = i + 1;
end
```

If Statement:

```
if x > 0 then
    disp("Positive number")
else
    disp("Non-positive number")
end
```

7. Plotting Functions

- Plot a function: `plot(x, y)`
- Multiple plots: `plot(x, y1, x, y2)`
- Labeling axes: `xlabel("X-axis"), ylabel("Y-axis")`
- Adding a title: `title("Graph Title")`

8. Numerical Integration & Differentiation

- Definite integral: `intg('x^2', 0, 5)`
- Derivative: `diff(y, x)`

Syntax

1. Cardinality of Sets

```
cardinality_A = length(A);
```

2. Inclusion-Exclusion Principle

```
Union_AB = union(A, B);
```

```
Intersection_AB = intersect(A, B);
```

```
cardinality_A = length(A);
```

```
cardinality_B = length(B);
```

```
cardinality_Intersection = length(Intersection_AB);
```

3. Evaluate a Polynomial at a Point

```
p = [2 -3 1]; x0 = 4;
```

```
value = horner(p, x0);
```

4. First-Order and Second-Order Differentiation

```
deff('y=f(x)', 'y=x^3 + 6*x^2 + 11*x + 6');
```

```
function df = first_derivative(f, x, h)
```

```
df = round((f(x + h) - f(x - h)) / (2 * h));
```

```
endfunction
```

```
function d2f = second_derivative(f, x, h)
```

```
d2f = round((f(x + h) - 2*f(x) + f(x - h)) / (h^2));
```

```
endfunction
```

5. Single, Double, Triple Integrals

```
function c=f(x)
```

```
c=x^2
```

```
endfunction
```

```
integral1 = intg(a,b,f);
```

6. Permutations and Combinations

```
P = factorial(n) / factorial(n - r);
```

```
C = factorial(n) / (factorial(r) * factorial(n - r));
```

7. Sum Rule and Product Rule Principle

```
union_AB = union(A, B);
```

```
total_ways = n1 * n2;
```

8. Conditional Probability

$$P_A = 0.6;$$

$$P_B = 0.5;$$

$$P_{AB} = 0.3;$$

$$P_{A_given_B} = P_{AB} / P_B;$$

9. 2D and 3D Graph Plotting

```
plot(x, y);
```

```
xlabel("x");
```

```
ylabel("y = x^2");
```

```
title("2D Plot of y = x^2");
```

```
[x, y] = meshgrid(-5:0.5:5, -5:0.5:5);
```

```
z = x.^2 + y.^2;
```

```
surf(x, y, z);
```

```
xlabel("x");
```

```
ylabel("y");
```

```
zlabel("z = x^2 + y^2");
```

```
title("3D Surface Plot");
```

10. Matrix Addition and Multiplication

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix};$$

$$B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix};$$

$$\text{sum_matrix} = A + B;$$

$$\text{prod_matrix} = A * B;$$

Problem 1:

Write a Scilab program to find cardinality of the given sets.

1. $A = \{1,3,5,7,9\}$
2. $B = \{x, y, z\}$ where $x = 1, y = 2, z = 3$
3. $C = \{Apple, Acer, Sony, Dell, Lenovo\}$ where Apple=1, Acer=2, Sony=3, Dell=4, Lenovo=5;
4. $D = \{1,1,2,3,4,5,6,6,7\}$

Aim: To write a Scilab program to determine the cardinality of given sets.

Program:

```
clear; clc; // Clear the console & values
A=[1,3,5,7,9];
a=length(A);
disp("Cardinality of set A is:" +string(a)) // '+ string' is to show the output on the same line
x=1,y=2,z=3;
B=[x,y,z];
b=length(B);
disp('Cardinality of set B is:'+string(b));
Apple =1, Acer=2, Sony=3, Dell=4, Lenovo=5;
C=[Apple, Acer, Sony, Dell, Lenovo];
c=length(C);
disp('Cardinality of set C is:'+string(c));
D=[1,1,2,3,4,5,6,6,7];
d=length(unique(D)); //unique command helps to eliminate the same elements
disp('Cardinality of set D is:'+string(d));
```

Output:

```
"Cardinality of set A is:5"
"Cardinality of set B is:3"
"Cardinality of set C is:5"
"Cardinality of set D is:7"
```

Problem 2:

Out of 1200 students of a college, 552 took Economics, 627 took Mathematics, 540 took Information Technology, 217 took Economics and Mathematics, 307 took Economics and Information Technology, 240 took Mathematics and Information Technology, 213 took all the three subjects. Using the inclusion-exclusion principle, write a Scilab program to determine:

- (i) The number of students who took at least one of the three subjects.
- (ii) The number of students who took none of the three subjects.

Aim: To write a scilab program to calculate the number of students taking at least one or none of the given subjects using inclusion-exclusion principle.

Program:

```
clear; clc; // Inclusion-Exclusion Principle in Scilab
U=1200; //total number of students
E=552; //number of students taken Economics
M=627; //number of students taken Mathematics
IT=540; //number of students taken Information Technology
EandM=217; //number of students taken Economics and Mathematics
EandIT=307; //number of students taken Economics and Information Technology
MandIT=240; //number of students taken Mathematics and Information Technology
EandMandIT=213; //number of students taken all the three subjects
//To find number of students taken at least one of the three subjects //By InclusionExclusion
Principle
EorMorIT=E+M+IT-EandM-EandIT-MandIT+EandMandIT;
disp('Number of students taken at least one of the three subjects is:' +string (EorMorIT));
//To find number of students not taken any of the three subjects
None=U-EorMorIT;
disp('Number of students not taken any of the three subjects is:' +string (None));
```

Output:

```
"Number of students taken at least one of the three subjects is:1168"
"Number of students not taken any of the three subjects is:32"
```

Problem 3:

Write a Scilab program to evaluate the following polynomials at the given value of x .

- i. $f(x) = x^3 - 2x + 1$ at $x = 2$.
- ii. $g(x) = x^2 - 1$ at $x = 3$.
- iii. $h(x) = 2x^3 - 7x^2 + 4x - 15$ at $x = 5$.

Aim: To write a Scilab program to evaluate these polynomials at the given value of x .

Program:

```
clc; clear;
// Define the polynomial f using its roots
f = poly([1, -2, 0, 1], "x", "c");
disp("The polynomial f is " + string(f)); // Evaluate polynomial f at x = 2
k = horner(f, 2);
disp("Value of polynomial f at x = 2 is " + string(k));
// Define the polynomial g
g = poly([-1, 0, 1], "x", "c");
disp("The polynomial g is " + string(g)); // Evaluate polynomial g at x = 3
l = horner(g, 3);
disp("Value of polynomial g at x = 3 is " + string(l));
// Define the polynomial h
h = poly([-15, 4, -7, 2], "x", "c");
disp("The polynomial h is " + string(h)); // Evaluate polynomial h at x = 5
m = horner(h, 5);
disp("Value of the polynomial h at x = 5 is " + string(m));
```

Output:

```
"the polynomial f is 1 -2x +x^3"
"value of polynomial f at x = 2 is 5"
"the polynomial g is -1 +x^2"
"value of polynomial g at x = 3 is 8"
"the polynomial h is -15 +4x -7x^2 +2x^3"
"value of the polynomial h at x = 5 is 80"
```

Problem 4:

Write a Scilab program to compute the first-order and second-order derivatives of the function $f(x) = x^3 + 6x^2 + 11x + 6$ at $x = 1$.

Aim: To write a Scilab program to compute first order and second order derivatives of a given function at a specific point.

Program:

```
clear; clc; // Define the function
deff('y=f(x)', 'y=x^3 + 6*x^2 + 11*x + 6');
// Define the first derivative using finite differences
function df = first_derivative(f, x, h)
df = round((f(x + h) - f(x - h)) / (2 * h)); // Central difference formula with rounding
endfunction
// Define the second derivative using finite differences
function d2f = second_derivative(f, x, h)
d2f = round((f(x + h) - 2*f(x) + f(x - h)) / (h^2)); // Second-order central difference with rounding
endfunction
// Define x-value where you want to compute derivatives
x0 = 1; // Chosen to ensure whole-number derivatives
h = 1e-5; // Small step size for numerical differentiation
// Compute derivatives
df_value = first_derivative(f, x0, h);
d2f_value = second_derivative(f, x0, h);
// Display results
disp("First derivative at x0:" + string(df_value));
disp("Second derivative at x0:" + string(d2f_value));
```

Output:

```
"First derivative at x0:26"
"Second derivative at x0:18"
```

Problem 5:

Write a Scilab program to compute the single integral, double integral and triple integral of function $f(x) = x^2$. The upper and lower limits for integration should be entered by user.

Aim: To write a Scilab program to compute single integral, double integral and triple integral of a given function with user defined limits.

Program:

```
clear; clc;

function c=f(x)
    c=x^2
endfunction

a=input("Enter the initial limit: ");
b=input("Enter the final limit: ");
// First Order Integral
integral1 = intg(a,b,f);
disp("First Order Integral: " + string(integral1));
function c=f2(x) // Integral of f(x) once
    c = intg(a,x,f); // Integrate f(x) from 'a' to 'x' // Second Order Integral (Integrating f(x) twice)
endfunction
integral2 = intg(a,b,f2);
disp("Second Order Integral: " + string(integral2));
function c=f3(x) // Integral of f2(x) once (i.e., integral of f(x) twice) // Third Order Integral
    (Integrating f(x) thrice)
    c = intg(a,x,f2);
endfunction
integral3 = round(intg(a,b,f3));
disp("Third Order Integral: " + string(integral3));
```

Output:

```
Enter the initial limit: 4
Enter the final limit: 8

"First Order Integral: 149.33333"
"Second Order Integral: 234.66667"
"Third Order Integral: 273"
```

Problem 6:

Write a Scilab program to compute permutations and combinations for the values entered by the user.

Aim: To write a Scilab program to compute permutations and combinations based on user input.

Program:

```
clear; clc;

// Function to calculate permutation (nPr)
function p=permutation(n, r)
    if n < r then
        error("n must be greater than or equal to r for permutations.");
    end
    p = factorial(n) / factorial(n - r);
endfunction

// Function to calculate combination (nCr)
function c=combination(n, r)
    if n < r then
        error("n must be greater than or equal to r for combinations.");
    end
    if r < 0 then
        error("r must be non-negative for combinations.");
    end
    c = factorial(n) / (factorial(r) * factorial(n - r));
endfunction

disp("Note: n value must be greater than or equal to r");
// Input values
n = input("Enter the value of n: ");
r = input("Enter the value of r: ");

// Calculate and display results
p_value = permutation(n, r);
c_value = combination(n, r);
```

```
disp("Permutation (nPr): " + string(p_value));  
disp("Combination (nCr): " + string(c_value));
```

Output:

If proper value is entered:

```
"Note: n value must be greater than or equal to r"  
Enter the value of n: 4  
  
Enter the value of r: 3  
  
"Permutation (nPr): 24"  
"Combination (nCr): 4"
```

If proper value is not entered:

```
"Note: n value must be greater than or equal to r"  
Enter the value of n: 5  
  
Enter the value of r: 8  
n must be greater than or equal to r for permutations.
```

Problem 7:

Suppose a bookcase shelf has 6 Mathematics textbooks, 5 Programming textbooks and 3 Networking textbooks.

- (i) Find the number of ways a student can choose a textbook from the shelf.
- (ii) Find the number of ways a student can choose from each category.

Write a Scilab program to discuss sum rule and product rule principles in probability.

Aim: To write a Scilab program to apply sum rule and product rule principles in probability.

Program:

```
//Scilab Program to Discuss Sum Rule and Product Rule in Probability
```

```
clc;
```

```
clear;
```

```
M=6; //number of mathematics textbook
```

```
P=5; //number of programming textbook
```

```
N=3; //number of networking textbook
```

```
T=M+P+N;
```

```
disp('Number of ways a student can choose a textbook:'+string(T));
```

```
A=M*P*N;
```

```
disp('Number of ways a student can choose a textbook of each type:'+string(A));
```

Output:

```
"Number of ways a student can choose a textbook:14"
```

```
"Number of ways a student can choose a textbook of each type:90"
```

Problem 8:

A pair of fair dice is thrown. Find the probability that the sum is 10 or greater in the following cases:

(a) A 5 appears on the first die.

(b) A 5 appears on at least one die.

Write a Scilab program to compute conditional probability based on the given cases.

Aim: To write a Scilab program to compute conditional probability for a given events.

Program:

```
clc; clear;
```

```
E1=['(5,1)','(5,2)','(5,3)','(5,4)','(5,5)','(5,6)']; //Outputs with 5 appears on the first die
```

```
A=['(4,6)','(5,5)','(5,6)','(6,4)','(6,5)','(6,6)']; //Outputs with sum of the outcomes are 10 or greater
```

```
AandE1=intersect(A,E1);
```

```
P1=2/6;
```

```
disp('Probability of event that sum of the outcomes are 10 or greater if 5 appears on the first  
die:'+string(P1));
```

```
E2=['(5,1)','(5,2)','(5,3)','(5,4)','(5,5)','(5,6)','(1,5)','(2,5)','(3,5)','(4,5)','(6,5)'];
```

```
AandE2=intersect(A,E2);
```

```
P2=3/11;
```

```
disp('Probability of event that sum of the outcomes are 10 or greater if 5 appears on at least one  
die:'+string(P2));
```

Output:

```
"Probability of event that sum of the outcomes are 10 or greater if 5 appears on the first die:0.3333333"  
"Probability of event that sum of the outcomes are 10 or greater if 5 appears on at least one die:0.2727273"
```

Problem 9:

Write a Scilab program to

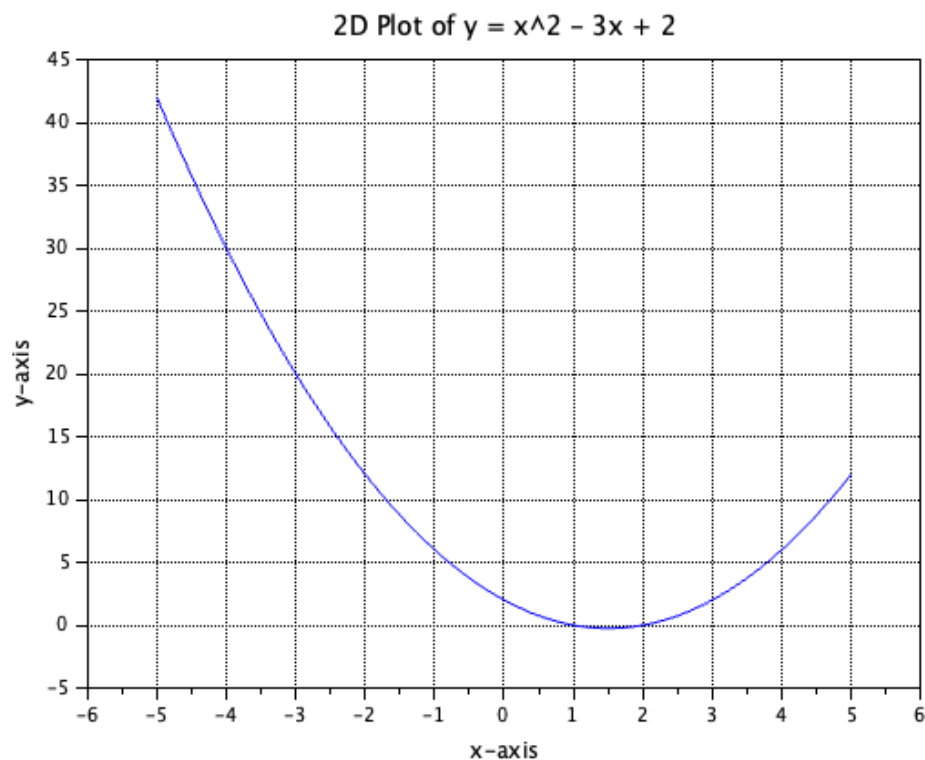
(i) plot a 2D graph of the quadratic equation $y = x^2 - 3x + 2$.

(ii) plot a 3D graph of the equation $z = x^2 + y^2$.

Aim: To write a Scilab program to plot 2D and 3D graphs of mathematical equations.

Program:

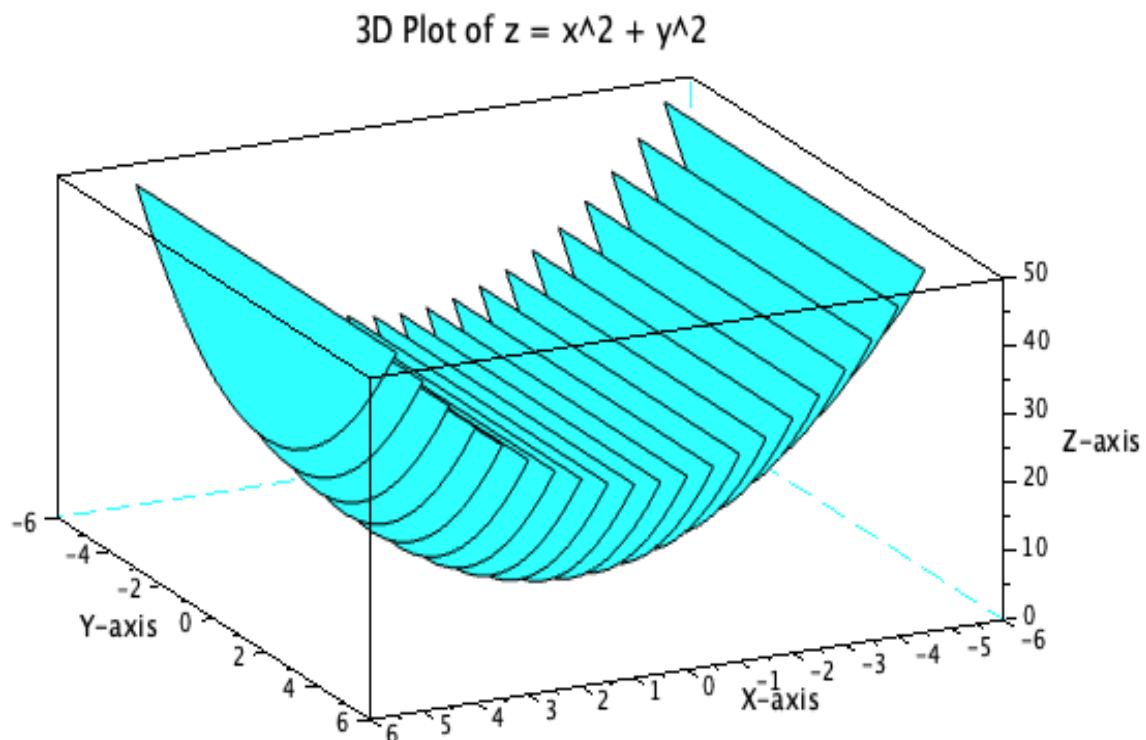
```
// 2D Plot of a Quadratic Equation  $y = x^2 - 3x + 2$   
clc; clear; close; // Clear console, variables, and figures  
x = linspace(-5, 5, 100); // Generate 100 points from -5 to 5  
y = x.^2 - 3*x + 2; // Define the quadratic equation  
plot(x, y, "b"); // Plot with blue color  
xlabel("x-axis"); ylabel("y-axis");  
title("2D Plot of  $y = x^2 - 3x + 2$ ");  
xgrid();
```

Output:

3D Plotting

```
// Define x and y ranges
x = linspace(-5, 5, 50);
y = linspace(-5, 5, 50);
// Create a meshgrid (for 3D plots)
[X, Y] = meshgrid(x, y);
// 3D Plot of a Simple Polynomial Surface
clc; clear; close;
[x, y] = meshgrid(-5:0.5:5, -5:0.5:5); // Create a grid from -5 to 5
z = x.^2 + y.^2; // Define the function z = x^2 + y^2
// Plot 3D surface
clf;
plot3d(x, y, z);
xlabel("X-axis");
ylabel("Y-axis");
zlabel("Z-axis");
title("3D Surface Plot of z = x^2 + y^2");
```

Output:



Problem 10:

Write a Scilab program to perform matrix addition and multiplication for the given matrices:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \text{ \& } B = \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}.$$

Aim: To write a Scilab program to compute matrix addition and multiplication.

Program:

```
// Define two matrices
clear; clc;
A = [1 2 3; 4 5 6; 7 8 9];
B = [9 8 7; 6 5 4; 3 2 1];
disp('Matrix A=',A);
disp('Matrix B=',B);
C = A + B; // Compute matrix addition
disp("Matrix Addition Result:A+B=");
disp(C);
D = A * B; // Compute matrix multiplication
disp("Matrix Multiplication Result:A*B=");
disp(D);
```

Output:

```
"Matrix A="
 1.  2.  3.
 4.  5.  6.
 7.  8.  9.
"Matrix B="
 9.  8.  7.
 6.  5.  4.
 3.  2.  1.
"Matrix Addition Result:A+B="
10. 10. 10.
10. 10. 10.
10. 10. 10.
"Matrix Multiplication Result:A*B="
30. 24. 18.
84. 69. 54.
138. 114. 90.
```