RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

DEPARTMENT OF MATHEMATICS

Number Theory, Vector Calculus

&

Computational Methods

MA221TC

MANUAL FOR EXPERIENTIAL LEARNING USING MATLAB

II - SEMESTER

Contents

Introduction

In Experiential Learning component of first semester Mathematics course - Multivariable Calculus, an introduction was given to MATLAB software and modules were provided for basic concepts like arithmetic operations, math built-in functions, visualizing 2D and 3D plots, operations of differential, integral and vector calculus, and evaluating multiple integrals.

On successful completion of learning these modules, the process continues with Experiential Learning component of second semester Mathematics course - Differential Equations & Numerical Methods. A note on scripts and anonymous functions appear which are useful in repeated usage of a set of MATLAB commands. Further a more rigorous approach is followed in which the modules cover all the basic concepts of higher Engineering Mathematics like evaluating handling statistical data, solving system of equation, solving higher order differential equations and implementing numerical techniques on MATLAB.

It is ensured that after going through the MATLAB course as Experiential Learning of Engineering Mathematics I and II, a student is well equipped to code any engineering problem successfully.

Scripts and Functions

MATLAB Scripts

To perform a sequence of commands repeatedly or to save for future reference, they are stored in a program file. The simplest type of MATLAB program is a script, which contains a set of commands exactly as it is typed at the command line.

Scripts are the simplest type of program file. They are useful for automating a series of MATLAB commands, such as computations that are performed repeatedly from the command line or series of commands referenced.

The new scripts are created in the following ways:

1. Click the **New Script** button on the **Home** tab.
2. Use the `edit` function by typing it in the MATLAB Command Prompt.

This opens the MATLAB Editor window in which commands can be entered. After entering the commands, save the script with a filename. The default MATLAB Script extension is '.m'.

After saving the script, the code can be run in the following ways:

1. Type the script name on the command line and press **Enter**. For example, to run a script titled `myMATLABProgram.m` script, type `myMATLABProgram`.
2. Click the **Run** button on the **Editor** tab.

Example for a MATLAB Script

The following script calculates the area of a triangle with base 'b' and height 'h'.

```
b = 5;
h = 3;
a = 0.5*(b*h)
```

This can be saved as `triarea.m`, run the script, by typing `triarea` on the MATLAB Command Prompt or click the **Run** button on the **Editor** tab.

Entering Comments in a MATLAB Script

Writing the comment lines in MATLAB script will help the user to understand the code.

Add comments to MATLAB code using the percentage (`%`) symbol. Comment lines can appear anywhere in a program file and comments can be appended at the end of a line of code.

```
%This script calculates the area of a triangle with base 'b' and height 'h'
b = 5;            % Base
h = 3;            % Height
a = 0.5*(b*h)     % Area
```

To obtain more information on Scripts, type the following in Command Window:

```
web(fullfile(docroot, 'matlab/scripts.html'))
```

Live Scripts

To combine code with embedded output, formatted text, equations and images in a single interactive environment, create live scripts.

Live scripts are program files useful for interacting with a series of MATLAB commands and their output. Live scripts contain output and graphics with the code that produced them, together in a single interactive environment called the Live Editor. Add formatted text, images, hyperlinks and equations to live scripts to produce an interactive narrative that can be shared with others. Live scripts are stored as files `.mlx` extension.

To create a new live script, on the **Home** tab, in the **New** drop-down menu, select **Live Script**.

To obtain more information on Live Scripts, type the following in the Command Window:

```
web(fullfile(docroot, 'matlab/matlab_prog/create-live-scripts.html'))
```

Valid script names follow the same rules as variable names. It must start with a letter and can be followed by letters, digits, or underscores.

MATLAB Functions

Both scripts and functions allow to reuse sequences of commands by storing them in program files. Scripts are the simplest type of program, since they store commands exactly as it is typed at the command line.

Functions provide more flexibility, as primarily user can pass input values and return output values. For example, the following function named `fact` computes the factorial of a number (`n`) and returns the result (`f`).

```
function f = fact(n)
    f = prod(1:n);
end
```

This type of function must be defined within a file, not at the command line. The best practice is to use the same name for the function and the file (in this example, `fact.m`), since MATLAB associates the program with the file name.

The function can be called from the command line, using the same syntax rules that apply to functions installed with MATLAB. For instances, calculate the factorial of 5.

```
x = 5;
y = fact(x);
```

Creating MATLAB Functions

Defining Functions

To create a MATLAB function, open the MATLAB Editor by typing `edit` in the Command Window.

The first line of every function is the definition statement, which includes the following elements.

1. The `function` keyword (required)
2. Output arguments (optional)
3. Function Name (required)
4. Input arguments (optional)

Valid function names follow the same rules as variable names. It must start with a letter and can be followed by letters, digits, or underscores.

```
function [x,y] = myFunction(one,two,three)
```

This above function has the following elements:

1. Function Name - `myFunction`
2. Input Arguments - `one, two, three`
3. Output Argument - `x, y`

Contents of Functions and Files

The body of a function can include valid MATLAB expressions, control flow statements, comments, blank lines and nested functions. Note that any variables can be created within a function and are stored within a workspace specific to that function, which is separate from the base MATLAB workspace.

End Statements

Typically, functions end with an `end` statement at the end of the file. Although it is optional, use `end` for better code readability.

Scripts vs Functions

For a comparison on scripts and functions and when to use them, go over the documentation in:

```
web(fullfile(docroot, 'matlab/matlab_prog/scripts-and-functions.html'))
```

Anonymous Functions

What Are Anonymous Functions?

An anonymous function is a function that is *not* stored in a program file. It is associated with a variable whose data type is `function_handle`. Anonymous functions can accept inputs and return outputs, just as standard functions do. However, they can contain only a single executable statement.

For example, create a `function_handle` to an anonymous function that finds the square of a number:

```
sqr = @(x) x.^2;
```

Variable `sqr` is a function handle. The `@` operator creates the handle and the parentheses `()` immediately after the `@` operator include the function input arguments. This anonymous function accepts a single input `x` and implicitly returns a single output, an array the same size as `x` that contains the squared values.

Find the square of a particular value (`5`) by passing the value to the function handle, similar to passing an input argument to a standard function.

```
a = sqr(5)
```

Many MATLAB functions accept function handles as inputs so that functions can be evaluated over a range of values. Handles can be created either for anonymous functions or for functions in program files. The benefit of using anonymous functions is that not to edit and maintain a file for a function that requires only a brief definition.

Example of Anonymous Function with ODE45 solver

The van der Pol equation is a second order ODE

$$y_1'' - \mu\left(1 - y_1^2\right) y_1' + y_1 = 0,$$

where $\mu > 0$ is a scalar parameter. Rewrite this equation as a system of first-order ODEs by making the substitution $y_1' = y_2$. The resulting system of first-order ODEs is

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= \mu(1 - y_1^2)y_2 - y_1. \end{aligned}$$

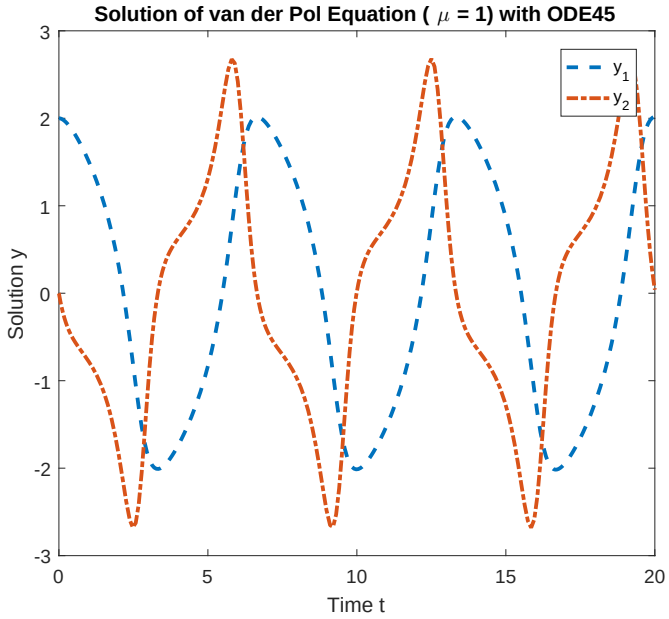
The function `vdpeval` represents the van der Pol equation using $\mu = 1$. The variables y_1 and y_2 are the entries `y(1)` and `y(2)` of a two-element vector, `dydt`.

Solve the ODE using the `ode45` function on the time interval `[0 20]` with initial values `[2 0]`. The resulting output is a column vector of time points `t` and a solution array `y`. Each row in `y` corresponds to a time returned in the corresponding row of `t`. The first column of `y` corresponds to y_1 and the second column to y_2.

```
[t,y] = ode45(@vdpeval,[0 20],[2; 0]);
```

Plot the solutions for y_1 and y_2 against `t`.

```
plot(t,y(:,1),'--',t,y(:,2),'-.')
title('Solution of van der Pol Equation (\mu = 1) with ODE45');
xlabel('Time t');
ylabel('Solution y');
legend('y_1','y_2')
```




```
function dydt = vdpeval(t,y)
%VDP1  Evaluate the van der Pol ODEs for mu = 1
%   Jacek Kierzenka and Lawrence F. Shampine
dydt = [y(2); (1-y(1)^2)*y(2)-y(1)];
end
```

Modules

1 Vector Differentiation

Topic learning outcomes:

Student will be able to:

1. Understand the existence of vector functions, derivatives of vector functions and rules of differentiation and fundamentals of the integration of vector point function.
2. The importance of defining vector differential operator and the operations- Gradient of scalar point functions, Divergence and Curl of vector point functions.

Syntax and description:

- `g = gradient(f,v)` returns the gradient vector of the scalar function f with respect to vector v in Cartesian coordinates. The input f is a function of symbolic scalar variables and the vector v specifies the scalar differentiation variables.
- `g = gradient(f)` returns the gradient vector of the scalar function f with respect to a vector constructed from all symbolic scalar variables found in f. The order of variables in this vector is defined by $symvar$.
- The gradient of a scalar function `f` with respect to the vector `v` is the vector of the first partial derivatives of `f` with respect to each element of `v`.
- `divergence(V,X)` returns the divergence of vector field V with respect to the vector X in Cartesian coordinates. Vectors V and X must have the same length.
- `curl(V,X)` returns the curl of the vector field V with respect to the vector X. The vector field V and the vector X are both three-dimensional.
- `curl(V)` returns the curl of the vector field V with respect to the vector of variables returned by $symvar(V, 3)$.

Example 1.1. *Find the gradient vector of* $2yzsin(x) + 3xsin(z)cos(y)$ *with respect to vector* `[x,y,z]`.

```
syms x y z
f(x,y,z) = 2*y*z*sin(x) + 3*x*sin(z)*cos(y);
gradient(f,[x,y,z])
```

$$ans(x, y, z) = \begin{pmatrix} 3\cos(y)\sin(z) + 2yz\cos(x) \\ 2z\sin(x) - 3x\sin(y)\sin(z) \\ 2y\sin(x) + 3x\cos(y)\cos(z) \end{pmatrix}$$

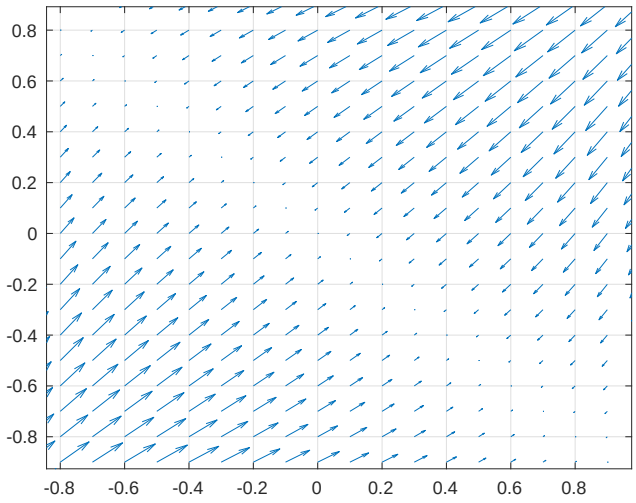
Example 1.2. *Find the gradient of a function* $-(sin(x) + sin(y))^2$ *with respect to vector* `[x,y]`., *and plot it as a quiver (velocity) plot*

```
syms x y
f = -(sin(x) + sin(y))^2;
g = gradient(f,[x,y])
```

$$g = \begin{pmatrix} -2\cos(x)\ (\sin(x)+\sin(y)) \\ -2\cos(y)\ (\sin(x)+\sin(y)) \end{pmatrix}$$

Now plotting the vector field defined by these components. MATLAB provides the `quiver` plotting function for this task. The function does not accept symbolic arguments. First, replace symbolic variables in expressions for components of `g` with numeric values. Then use `quiver`.

```
[X, Y] = meshgrid(-1:.1:1,-1:.1:1);
G1 = subs(g(1),[x y],{X,Y});
G2 = subs(g(2),[x y],{X,Y});
quiver(X,Y,G1,G2)
```



Example 1.3. *Find the divergence of the vector field* $V(x,y,z) = (x, 2y^2, 3z^3)$ *with respect to vector* $X = (x,y,z)$.

```
syms x y z
f = [x 2*y^2 3*z^3];
v = [x y z];
divergence(f,v)
```

$$ans = 9z^2 + 4y + 1$$

Example 1.4. *Find the divergence of the gradient of scalar function* $x^2 + y^2 + z^2$.

```
syms x y z
f = x^2 + y^2 + z^2;
divergence(gradient(f,vars),vars)
```

$$ans = 6$$

Example 1.5. *Compute the curl of the vector field* $V(x,y,z) = (x^3y^2z, y^3z^2x, z^3x^2y)$ *with respect to vector* $X = (x,y,z)$ *in Cartesian coordinates.*

```
syms x y z
V = [x^3*y^2*z, y^3*z^2*x, z^3*x^2*y];
X = [x y z];
curl(V,X)
```

$$ans = (x^2z^3 - 2xy^3z, x^3y^2 - 2xyz^3, -2x^3yz + y^3z^2)$$

Example 1.6. *Compute the curl of the gradient of the scalar function* $x^2 + y^2 + z^2$.

```
syms x y z
f = x^2 + y^2 + z^2;
vars = [x y z];
curl(gradient(f,vars),vars)
```

$$ans = (0, 0, 0)$$

Exercise:

1. Gradient of scalar measure the increase or decrease of a quantity along some direction, Find the gradient vector of $xy^2z^3 - x^3y^2z$ with respect to vector `[x,y,z]`.
2. The steepness of the slope at that point is given by $2x^3y^2z^4$. Find the divergence of the gradient of scalar function.
3. A person on a hang glider is spiraling upward due to rapidly rising air on a path having a vector field $F(x, y, z) = (\frac{x}{x+y}, \frac{y}{x+y})$. Find the divergence of the vector field with respect to vector $X = (x, y)$.
4. Show that the uniform motion of a projectile along a meridian of the rotating earth is given by the vector function $V(x, y, z) = (2xyz, xy - y^2z, x^2 - zx)$ is solenoidal.
5. The velocity vector field for the three-dimensional flow of an ideal fluid around a cylinder is given by $F(x, y, z) = (sinx + z, cosy - z, x - y)$. Show that $F(x, y, z)$ is irrotational.
6. The temperature of all points in a room at a particular time is a scalar field, then compute the curl of the gradient of the scalar function $x^2y + 2xy + z^2$.

2 Interpolation

Topic learning outcomes:

Student will be able to:

1. Interpolate from equally and unequally spaced domain points.
2. Extrapolate for evaluating points that lie outside the domain.
3. Use this concept to solve real life and engineering problems.

Syntax and description:

- `yq = interp1(x,y,xq)` - returns interpolated values of a 1-D function at specific query points using linear interpolation. `x` contains the sample points and `y` contains the corresponding values, $y(x)$. `xq` contains the coordinates of the query points.
- `yq = interp1(x,y,xq,method)` - specifies an alternative interpolation method: `'nearest'`, `'next'`, `'previous'`, `'linear'`,`'spline'`,`'pchip'`, or `'cubic'`. The default method is `'linear'`.
- `yq = interp1(x,y,xq,method,extrapolation)` - specifies a strategy for evaluating points that lie outside the domain of `x`. Set `extrapolation` to `'extrap'` when you want to use the `method` algorithm for extrapolation. Alternatively, you can specify a scalar value, in which case, `interp1` returns that value for all points outside the domain of `x`.

Interpolation method, specified as a value from the table below.

Method	Description
'linear'	Linear interpolation.
'nearest'	Nearest neighbor interpolation.
'next'	Next neighbor interpolation.
'previous'	Previous neighbor interpolation.
'pchip'	Shape-preserving piecewise cubic interpolation.
'cubic'	Same as 'pchip'.
'spline'	Spline interpolation using not-a-knot end conditions.

Example 2.1. *Using interpolation process find the value of* $\sin 48^o$.

Hint: Define the sample points x and corresponding sample values v.

```
x = 0:5:90;
v = sind(x);
xq = 48;
vq1 = interp1(x,v,xq)

vq1 =

    0.7425
```

Example 2.2. *The following table gives the distances (in miles) of the visible horizon for the given heights (in feet) above the earth's surface.*

x :	200	250	300	350	400
$y = f(x)$:	15.04	16.81	18.42	19.9	21.27

Find the value of y at $x = 218$ and 389 ft.

```
x = 200:50:400;
v = [15.04 16.81 18.42 19.9 21.27];
xq = [218 389];
vq1 = interp1(x,v,xq)
vq3 = interp1(x,v,xq,'spline')
```

```
vq1 =

   15.6772   20.9686


vq3 =

   15.6977   20.9771
```

Example 2.3. *The area A of a circle for different diameters d is given in the following table:*

d :	80	85	90	95	100
A :	5026	5674	6362	7088	7854

Calculate the area of a circle of diameter 105 units.

```
x=[80    85    90    95    100];
y= [5026    5674    6362    7088    7854];
xq=105;
yq1=interp1(x,y,xq,'pchip')
vq3 = interp1(x,y,xq,'spline')
```

```
yq1 =

   8.6589e+03


vq3 =

        8663
```

Example 2.4. *The following table gives the viscosity of oil as a function of temperature. Find the viscosity of oil at a temperature of* 140^o.

Temp(degree):	110	130	160	190
Viscosity:	10.8	8.1	5.5	4.8

```
x=[110    130    160    190];
y=[10.8    8.1    5.5    4.8];
xq= 140;
yq1 = interp1(x,y,xq,'spline')


yq1 =

    7.0333
```

Exercise:

1. Estimate the population in 1895 and 1925 from the following statistics:

Year x :	1891	1901	1911	1921	1931
Population y :	46	66	81	93	101

2. Using interpolation find the value of $e^{1.85}$.

3. The pressure p of wind corresponding to velocity v is given in following table. Estimate p when v = 25, 35, 45.

Year v :	10	20	30	40
Population p :	1.1	2	4.4	7.9

4. A rod is rotating in a plane. The following table gives the angle θ (radians) through which the rod has turned for various values of the time t second.

t :	0	0.2	0.6	0.8	1.2
θ :	0	0.12	1.12	2.02	4.67

Calculate the angle θ at t = 0.4, 1, 1.3.

3 Higher Order Ordinary Differential Equations

Topic learning outcomes:

Student will be able to:

1. Solve linear differential equations of second and higher order with constant coefficients.
2. Solve linear differential equations of second and higher order with variable coefficients.
3. Obtain solution of initial and boundary value problems.

Syntax and description:

- `S = dsolve(eqn)` - solves the ordinary differential equation `eqn`. Here `eqn` is a symbolic equation containing `diff` to indicate derivatives. Alternatively, you can use a string with the letter `D` indicating derivatives.
- `S = dsolve(eqn,cond)` - solves the ordinary differential equation `eqn` with the initial or boundary condition `cond`.

Example 3.1. *Solve* $2x^2\frac{d^2y}{dx^2} + 3x\frac{dy}{dx} - y = 0$.

```
syms y(x)
dsolve(2*x^2*diff(y, 2) + 3*x*diff(y) - y == 0)
```

```
ans =

C1/(3*x) + C2*x^(1/2)
```

Example 3.2. *Solve the boundary value problem* $y'' + 6y' + 9y = 2e^{-3x}$, $y(0) = 0$ *and* $y(1) = 1$.

```
syms y(x)
dsolve(diff(y, 2)+6*diff(y) +9*y == 2*exp(-3*x),y(0)==0,y(1)==1)
```

```
ans =

x^2*exp(-3*x) + x*exp(-3*x)*(exp(3) - 1)
```

Example 3.3. *Obtain the solution of the differential equation* $(D^4 - 1)y = \cos 2x \cosh x$.

```
syms y(x)
simplify(dsolve(diff(y, 4)-y == cos(2*x)*cosh(x)))
```

```
ans =

C5*cos(x) - (cos(2*x)*exp(x))/160 + C8*exp(x) - (3*sin(2*x)*exp(x))/160 ...
+ C6*sin(x) - (cos(2*x)*exp(-x))/160 + C7*exp(-x) + (3*sin(2*x)*exp(-x))/160
```

Example 3.4. *A 32 lb weight is suspended from a spring having constant 4 lb/ft. Find position of weight at any time, if a force* $16\sin 2t$ *is applied and damping force is negligible. Assume that initially the weight is at rest in the equilibrium position.*

Hint: The differential equation describing this phenomenon is

$$\frac{d^2x}{dt^2} + 4x = 16\sin 2t;\ x(0) = 0, x'(0) = 0.$$

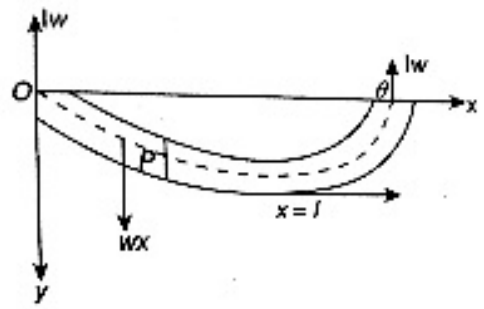
```
syms x(t)
Dx=diff(x);
simplify(dsolve(diff(x, 2)+4*x == 16*sin(2*t),x(0)==0,Dx(0)==0))
```

```
ans =

2*sin(2*t) - 4*t*cos(2*t)
```

Example 3.5. *A horizontal beam of length* $2lft$ *is freely supported at both ends. Find the maximum deflection if the load is* w *per unit length.*

Hint:



Let P be any point (x, y) on the elastic curve. The total weight supported is $2lw$. Hence the upward thrust of the support at each end is lw. Consider the segment OP of the beam. The forces acting in this region are upward thrust lw at zero and load wx acting at the midpoint of OP then the differential equation describing this phenomena is

$$El\frac{d^2y}{dx^2} = \frac{wx^2}{2} - wlx, \text{ under the boundary conditions } y(0) = 0, y(2l) = 0.$$

```
syms y(x) E l w
dsolve(E*l*diff(y, 2) == (w*x^2/2)-w*l*x, y(0) == 0,y(2*l) == 0)
max_deflection=subs(ans,x,l)


ans =

(x*(8*w*l^3 - 4*w*l*x^2 + w*x^3))/(24*E*l)
max_deflection =

(5*l^3*w)/(24*E)
```

Exercise:

1. Find the transient and steady state solutions of the mechanical system corresponding to the differential equation $\frac{d^2x}{dt^2} + 2\frac{dx}{dt} + 2x = \sin 2t - 2\cos 2t$; $x(0) = 0$, $\frac{dx}{dt}(0) = 0$.
2. In the case of a stretched elastic spring which has one end fixed and a particle of mass m attached at the other end, the equation of motion is $m\frac{d^2x}{dt^2} = -\frac{mg}{e}(x - l)$ where l is the natural length of the string and e is the elongation due to weight mg. Find x under the condition that at $t = 0$, $x = a$ and $\frac{dx}{dt} = 0$.
3. The deflection of a strut of length l with one end $(x = 0)$ built-in and the other supported subjected to end thrust p satisfies the equation $\frac{d^2y}{dx^2} + a^2y = \frac{a^2R}{p}(l - x), y = 0, \frac{dy}{dx} = 0$ at $x = 0$. Find the deflection.
4. An electric circuit consists of an inductance of 0.05 Henrys, a resistance of 5 Ohms and a condenser of capacitance 4×10^{-4} Farad and a constant emf of 110 Volts. Governing differential equation is $\frac{d^2Q}{dt^2} + 100\frac{dQ}{dt} + 50000Q = 2200$, under the conditions $Q(0) = I(0) = 0$. Determine charge $Q(t)$.

4 Number Theory

Topic learning outcomes:

Student will be able use built-in functions to:

1. Evaluate problems in number theory.
2. Compute divisors, GCD of integers.
3. Determine small primes.

Syntax and description:

- `divisors(n)`-finds all nonnegative divisors of an integer n.
- `divisors(expr,vars)`-finds the divisors of a polynomial expression `expr`. Here `vars` are polynomial variables.
- `G=gcd(A)`-finds the greatest common divisor of all elements of A.
- `G=gcd(A,B)` - finds the greatest common divisor of A and B.
- `nextprime(n)` - returns the next prime number greater than or equal to `n`. If `n` is a vector or matrix, then `nextprime` acts element-wise on `n`.
- `nthprime(n)`-returns the nth prime number. `nthprime` acts element-wise on array inputs.
- `prevprime(n)`-returns the largest prime number smaller than or equal to `n`.
- `TF = isprime(X)` returns a logical array the same size as `X`. The value at `TF(i)` is true when `X(i)` is a prime number. Otherwise, the value is false.
- `rem(a,b)`- finds the remainder after division of `a` by `b`.
- `mod(a,m)`-returns the modulus after division of `a` by `m`.
- `factor(n)` computes the prime factorization of a positive integer `n`.
- `eulerPhi(n)` evaluates the Euler phi function or (also known as the totient function) for a positive integer `n`.
- `[Q,R]=quorem(A,B,var)` divides `A` by `B` and returns the quotient `Q` and remainder R of the division, such that `A = Q*B + R`. This syntax regards `A` and `B` as polynomials in the variable `var`.

Example 4.1. *Find the divisors of integer 42.*

```
>> divisors(42)


ans =
1     2     3     6     7    14    21    42
```

Example 4.2. *Calculate the greatest common divisor of the integers* $4420, -128, 8984, -488$.

```
>> A = sym([4420, -128, 8984, -488])
>> gcd(A)
```

```
A =
[4420, -128, 8984, -488]
ans =
4
```

Example 4.3. *Obtain the next prime number greater than 123.*

```
>> nextprime(123)


ans =

127
```

Example 4.4. *Find the 153rd prime number.*

```
>> nthprime(153)


ans =

883
```

Example 4.5. *Determine the 10th, 100th, and 1000th prime numbers.*

```
>> n = [10 100 1000];
>> nthprime(n)


ans =

29         541        7919
```

Example 4.6. *Find the largest prime number smaller than 82.*

```
>> prevprime(82)


ans =

79
```

Example 4.7. *Check whether the following numbers are prime.*

$$12,\ 129,\ 137,\ 1399.$$

```
>> A = [12, 129, 137, 1399];
>> T = isprime(A)
```

```
T =

1x4 logical array

0   0   1   1
```

Example 4.8. *Evaluate the remainder after division of the number 27 by 4.*

```
>> rem(27, 4),


ans =

3
```

Example 4.9. *Obtain 87 congruent modulo 5.*

```
>> mod(87, 5)


ans =

2
```

Example 4.10. *Calculate the prime factorization of 120.*

```
>> factor(120)


ans =

2     2     2     3     5
```

Example 4.11. *Calculate Euler function $\phi(n)$ of the integer 120.*

```
>> eulerPhi(120)


ans =

40
```

Example 4.12. *Calculate Euler function $\phi(n)$ for the integers 12, 29, 120.*

```
>> A = [12, 29, 120];
>> eulerPhi(A)
```

```
ans =

4    28    32
```

Example 4.13. *Determine the remainder and quotient of the polynomials $x^3y^4 - 2xy + 5x + 1$, and xy*

```
syms x y
p1 = x^3*y^4 - 2*x*y + 5*x + 1;
p2 = x*y;
[q, r] = quorem(p1, p2, y)


q =

x^2*y^3 - 2

r =

5*x + 1
```

Exercise:

1. Find the greatest common divisor of the integers 20, 50 and 120.
2. Determine the divisors of integer 129, 251, 476.
3. Calculate the sum of the positive divisors of 20, 85, 164.
4. Obtain the largest prime number smaller than 32, 149, 2019.
5. Compute the quotient and remainder of the polynomials $x^5 + 7xy^2 + 6x + 1$ and x^2y.

5 Curve fitting, correlation and regression

Topic learning outcomes:

Student will be able sse built-in functions to:

1. Determine linear, quadratic and non-linear fits for a given data.
2. Determine correration and regression lines.

Syntax and description:

- `fitobject = fit(x,y,'fitType')` creates the fit to the data in `x` and `y` with the model specified by `fitType`.
- `[r,m,b] = regression(x,y)` returns correlation coefficient `r` between two variables `x` and `y`, slope `m` of the regression line of `y` on `x`, i.e, `y= mx+ b` and constant term `b`.

`'fitType'` are :

1. `'Poly1'` for Linear model :
$$f(x) = p1x + p2$$
2. `'Poly2'` for Quadratic model:
$$f(x) = p1x^2 + p2x + p3$$
3. `'Exp1'` for exponential:
$$f(x) = a\exp(bx)$$
4. `'Power1'` for power:
$$f(x) = ax^b$$

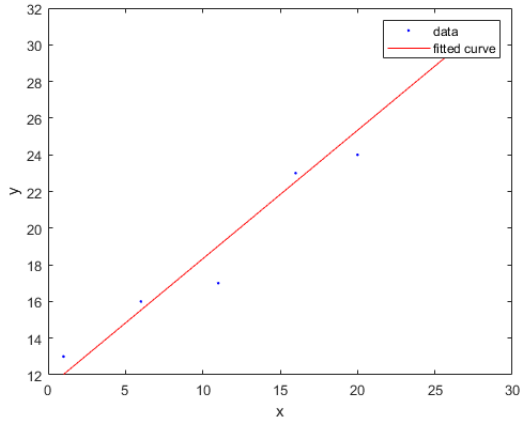
Example 5.1. *Fit a straight line to the following data.*

x	*1*	*6*	*11*	*16*	*20*	*26*
y	*13*	*16*	*17*	*23*	*24*	*31*

```
x = [1 6 11 16 20 26];
y =[13 16 17 23 24 31];
f=fit(x',y','Poly1')
plot(f,x,y)
```

```
f = Linear model Poly1:
f(x) = p1*x + p2
Coefficients (with 95% confidence bounds):
p1 =  0.7008  (0.4936, 0.908)
p2 =   11.32  (8.057, 14.59)
```

Example 5.2. *The following table gives the results of the measurements of train resistances; V is the velocity in mile per hour and R is the resistance in pound per ton.*

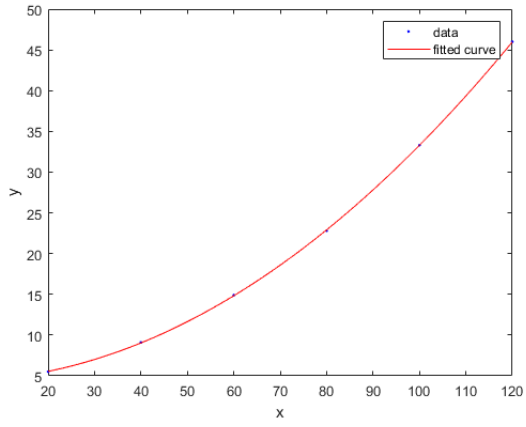


x	20	40	60	80	100	120
y	2.2	9.1	14.9	22.8	33.3	46

If R is related to V by the relation $R = a + bV + cV^2$.

```
x=[20 40 60 80 100 120];
y=[5.5 9.1 14.9 22.8 33.3 46];
f=fit(x',y','Poly2')
plot(f,x,y)
```

```
f = Linear model Poly2:
f(x) = p1*x^2 + p2*x + p3
Coefficients (with 95% confidence bounds):
p1 =    0.002871  (0.002753, 0.002989)
p2 =    0.002411  (-0.01447, 0.01929)
p3 =        4.35  (3.834, 4.866)
```



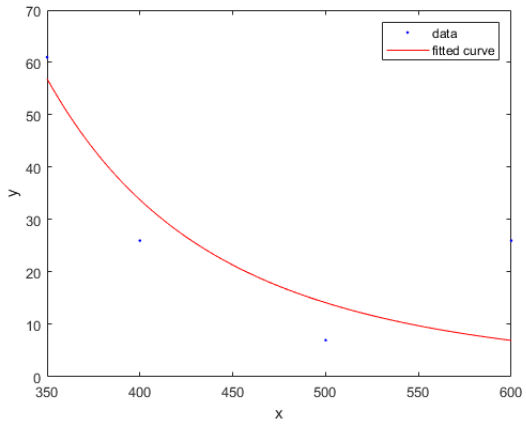
Example 5.3. *An experiment gave the following values,*

x	350	400	500	600
y	61	26	7	26

It is known that v and t are connected by the relation $v = at^b$.

```
v=[350  400 500 600]';
t=[61  26 7 26]';
f=fit(v,t,'Power1')
plot(f,v,t)
```

```
f = General model Power1:
f(x) = a*x^b
Coefficients (with 95% confidence bounds):
a =   4.802e+11  (-2.764e+13, 2.86e+13)
b =  -3.902  (-13.81, 6.005)
```

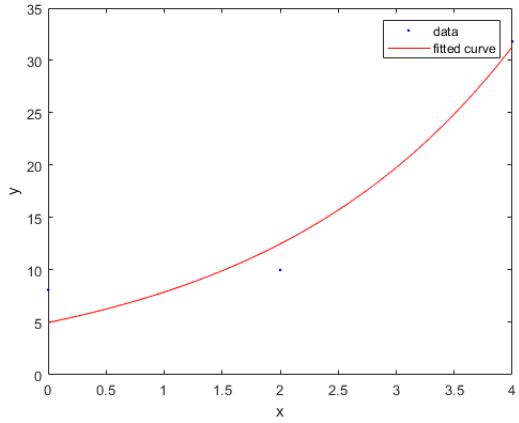


Example 5.4. *Fit a curve of the form $y = ae^{bx}$ to the data by the method of least squares.*

x	0	2	4
y	8.12	10	31.82

```
x=[0 2 4];
y=[8.12 10 31.82];
f=fit(x',y','Exp1')
plot(f,x,y)
```

```
f = General model Exp1:
f(x) = a*exp(b*x)
Coefficients (with 95% confidence bounds):
a =       4.989  (-27.4, 37.38)
b =      0.4593  (-1.273, 2.192)
```



Example 5.5. *The following table gives the stopping distance y in meters of a motor bike moving at a speed of x kms/hour when the breaks are applied.*

x	*16*	*24*	*32*	*40*	*48*	*56*
y	*0.39*	*0.75*	*1.23*	*1.91*	*2.77*	*3.81*

Find the correlation coefficient between the speed and the stopping distance, and the equations of regression lines.

```
x=[16, 24, 32, 40, 48, 56];
y=[0.39, 0.75, 1.23, 1.91, 2.77, 3.81];
[r,m,b] = regression(x',y')


r = 0.9827
m = 0.0851
b = -1.2551
```

Exercise:

1. Estimate the mean radiation dose at an altitude of 3000 ft by fitting a curve $y = ae^x$ to the given data:

x	50	450	780	1200	4400	4800	5300
y	28	30	32	36	51	58	69

2. The latent heat of vaporisation of steam r is given in the following table at different temperature t:

x	40	50	60	70	80	90	100	110
y	1069.1	1063.6	1058.2	1052.7	1049.3	1041.8	1036.3	1030.8

   Fit a relation to the form $r = a + bt$.

3. The following data represent carbon dioxide(co_2) emission from coal-fluid boilers (In units of 1000 tons) over a period of years 2005 to 2017. The variable (year) has been standardized and yield following table.

Years, x	0	5	8	9	10	11	12
co_2 emmision, y	910	680	520	450	370	350	340

Find regression model of y on x.

4. Obtain the lines of regression and hence find the coefficient of correlation for the following data

x	1	3	4	2	5	8	9	10	13	15
y	8	6	10	8	12	16	16	10	32	32

DEPARTMENT OF MATHEMATICS

VISION

Disseminate Scientific and Engineering knowledge through quality teaching and research partnership.

MISSION

- Develop a spirit of competence to face challenges of the rapidly changing teaching methodology.
- Motivate faculty and scholars to acquire the latest skills for mathematical modeling for modern technology application.
- Develop positive attitude towards continuous learning.
- Undertake inter-disciplinary research partnership.
- Impart quality education through effective teaching process.
- Provide extension programs for assisting individuals and organizations to find solutions to engineering problems through consultation and research.
- Impart skills on passing out-graduates to become excellent thereby enabling them to make significant contribution in their chosen profession.

Rashtreeya Sikshana Samithi Trust

RV COLLEGE OF ENGINEERING®

R. V. Vidyaniketan Post, Mysuru road

Tel: +91-80-67178021, 67178099 Fax: +91-80-67178011

www.rvce.edu.in