

MATLAB for Physics

Manual for BS Computer Science Students

Supervised by:

Dr. Saman Shahid
(Department of S&H, Lahore Campus)

Collaborators:

Muhammad Shiraz Ahmad
Muzamil Shah
Faizan Saleem
Shanian Mehar
Muhammad Bilal Azam



Department of Science and Humanities
National University of Computer and Emerging Sciences, Lahore Campus.

Contents

List of Figures	3
1 Introduction to MATLAB	4
1.1 Some Basic Commands	4
1.1.1 Vectors and Matrices	4
1.1.2 Basic Functions	5
1.1.3 Plotting in MATLAB	7
1.1.4 Element–Wise Operations	11
1.2 Vectors, Arrays and Matrices	12
1.2.1 Vector Algebra	12
1.3 Loops	15
1.3.1 For Loop	15
1.3.2 If Else Condition	16
2 Physics with MATLAB	17
2.1 Mechanics	17
2.1.1 Gravitational Force between Two Masses	17
2.1.2 Gravitational Force–An Inverse Square Law	17
2.1.3 Free Fall Motion	18
2.1.4 Projectile Motion without making Custom Functions	19
2.1.5 Projectile Motion by making Custom Functions	22
2.2 Waves and Oscillations	24
2.2.1 SHM as Circular Motion	24
2.3 Electricity and Magnetism	30
2.3.1 Coulomb Force between Charges	30
2.3.2 Coulomb Force between Charges using For Loop	31
2.3.3 Fields due to Discrete and Line Charge Distributions	33
2.3.4 Electric Fields due to dipole in a 2D plane using the Coulomb’s Law	36
2.3.5 Electric Field due to a Point Charge	39
2.3.6 Electric Field due to a Dipole	39
2.3.7 Electric Potential due to a Point Charge	40
3 Some Structured Questions with Numerical Analysis	42
3.1 Questions from Vector Algebra	42
3.2 Questions from One–Dimensional Motion	45

List of Figures

1.1	A sample 2D graph	7
1.2	A sample 3D graph	8
1.3	A sample mesh plot	9
1.4	A sample surf plot	9
1.5	A sample plot with title and axes labeling	10
1.6	A sample subplot	11
2.1	Falling objects in free fall (for $t = 11.3$ seconds)	19
2.2	Trajectory of a projectile	20
2.3	Range of a projectile	21
2.4	Height of a projectile	21
2.5	Trajectory of a projectile (using custom functions)	23
2.6	Phase Angle vs. Position	27
2.7	Phase Angle vs. Velocity	28
2.8	Phase Angle vs. Acceleration	28
2.9	SHM as circular motion	29
2.10	Phase Angle vs. Position, Velocity and Acceleration	29
2.11	Electrostatic (Coulomb) force as inverse square law	30
2.12	Electrostatic (Coulomb) force as inverse square law	33
2.13	Electric Field (Log Magnitude) due to point charge at origin (0,0,0)	35
2.14	Electric Field (Log Magnitude) due to line charge along z -axis	35
2.15	Electric field distribution of a dipole	38
2.16	Electric Field due to a Point Charge	39
2.17	Electric Field due to a Dipole	40
2.18	Electric Potential due to a Point Charge	41

Chapter 1

Introduction to MATLAB

1.1 Some Basic Commands

MATLAB uses double-precision floating point arithmetic accurate to approximately 15 digits, however, only 5 digits are displayed, by default. ¹ To display more digits, type `format long`. Then all subsequent numerical output will have 15 digits displayed. Type `format short` to return to 5-digit display.

```
1 >> pi
2 ans =
3     3.1416
4 >> format long
5 >> pi
6 ans =
7     3.141592653589793
8 >> format short
9 >> pi
10 ans =
11     3.1416
```

1.1.1 Vectors and Matrices

```
1 >> u = [1,5,11,4]           % Vector u is defined separated by commas
2 u =
3     1     5    11     4
4 >> v = [2 -1 6 -7 10 -7]    % Vector v is defined separated by spaces
5 v =
6     2    -1     6    -7    10    -7
7 >> newv = [u v]             % Concatenating vectors
```

¹Most of the content of this section is taken from the book *A Guide to MATLAB: For Beginners and Experienced Users* by Brian R. Hunt, Jonathan Rosenberg, and Ronald L Lipsman

```

8 newv =
9      1      5     11      4      2     -1      6     -7     10     -7
10 >> u4 = u(4)           % Extracting the elements of a vector, i.e., u
    (4)
11 u4 =
12      4
13 >> u=[1:6]             % Generate a vector of equally-spaced elements
    with colon operator
14 u =
15      1      2      3      4      5      6
16 >> u=[1:2:14]          % Increment by 2
17 u =
18      1      3      5      7      9     11     13
19 >> transu = u'          % Get the tranpose of u
20 transu =
21      1
22      3
23      5
24      7
25      9
26     11
27     13

```

To type a matrix you must: begin with a square bracket, separate elements in a row with commas or spaces, use a semicolon to separate rows, end the matrix with another square bracket.

```

1 >> A = [1 2 3; 4 5 6; 7 8 9]
2 A =
3      1      2      3
4      4      5      6
5      7      8      9
6 >> A3r = A(3,:)        % To publish 3rd row of the a matrix
7 A3r =
8      7      8      9
9 >> A2c = A(:,2)        % To publish 2nd column of the a matrix
10 A2c =
11      2
12      5
13      8

```

1.1.2 Basic Functions

In MATLAB you will use built-in functions as well as functions that you create yourself. MATLAB has many built-in functions, typing `help elfun` and/or `help specfun` calls up full lists of elementary and special functions. These include `sqrt`, `cos`, `sin`, `tan`, `log`, and `exp`.

```

1  >> a = sin(45)           % Compute sine of 45 in radians
2  a =
3      0.8509
4  >> b = sind(45)         % Compute sine of 45 in degrees
5  b =
6      0.7071
7  >> c = cos(45)           % Compute cosine of 45 in radians
8  c =
9      0.5253
10 >> d = cosd(45)          % Compute cosine of 45 in degrees
11 d =
12     0.7071
13 >> e = tan(45)           % Compute tangent of 45 in radians
14 e =
15     1.6198
16 >> f = tand(45)          % Compute tangent of 45 in degrees
17 f =
18     1
19 >> g = acsc(45)          % Compute the cosecant-inverse of 45 in radians
20 g =
21     0.0222
22 >> h = asec(45)          % Compute the secant-inverse of 45 in radians
23 h =
24     1.5486
25 >> i = acotd(45)         % Compute the cotangent-inverse of 45 in degrees
26 i =
27     1.2730
28 >> j = log(45)           % Compute the natural logarithm of 45
29 j =
30     3.8067
31 >> k = log10(45)         % Compute the common logarithm of 45
32 k =
33     1.6532
34 >> l = log2(45)          % Compute the logarithm in base 2 of 45
35 l =
36     5.4919
37 >> m = exp(45)           % Compute the exponential of 45
38 m =
39     3.4934e+19
40 >> n = sqrt(45)          % Compute the square root of 45
41 n =
42     6.7082

```

²Major content of this section taken from the <https://www.mathworks.com/help/index.html>.

1.1.3 Plotting in MATLAB

The command `plot` produces 2D graphics². Before using `plot` command, define the interval for the independent variable x and the function of the form $y = f(x)$. Then `plot(x,y)` command is called to obtain the figure of $f(x)$ with respect to x , as shown in figure 1.1.

```
1 >> x = 0:0.01:2*pi;  
2 >> y = sin(x);  
3 >> plot(x,y)
```

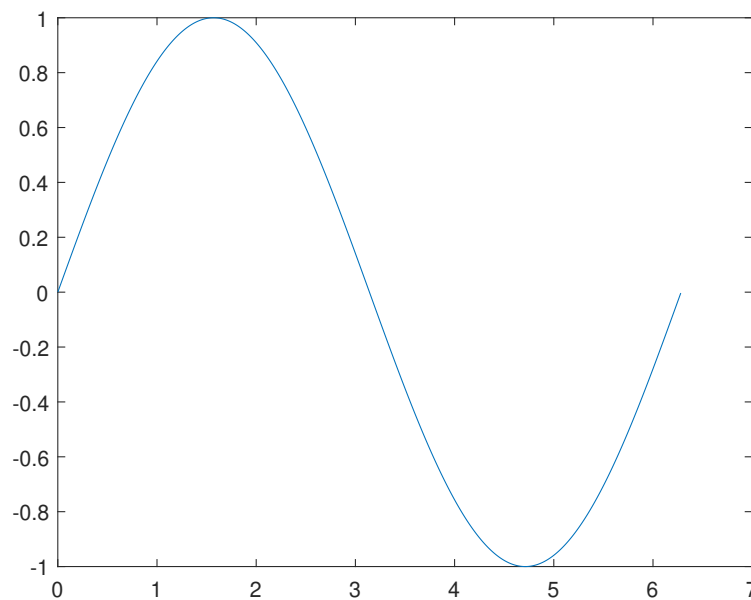


Figure 1.1: A sample 2D graph

And for 3D plots, use `plot3(x,y,z)`. A graph of sample function is shown in figure 1.2.

```
1 >> x = 1:5;  
2 >> y = [0 -3 -5 12 3];  
3 >> z = 2:2:10;  
4 >> plot3(x,y,z, 'k*')  
5 >> grid
```

MATLAB has several other plotting functions: `fplot` (similar to `plt`), `subplot` (multiple plots on the same window), `ezplot3` (3D plots), `mesh` (3D plots), `surf` (3D plots) etc. For example, to create `mesh`, assume we have three matrices of the same size. Then plot them as a mesh plot. The plot uses Z for both height and color. It is shown in figure 1.3.

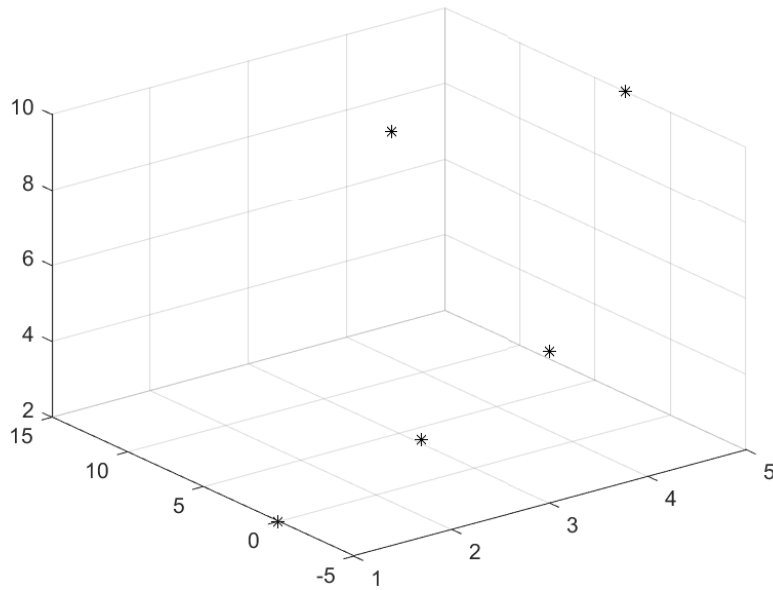


Figure 1.2: A sample 3D graph

```

1 [X,Y] = meshgrid(-8:.5:8);
2 R = sqrt(X.^2 + Y.^2) + eps;
3 Z = sin(R)./R;
4 mesh(X,Y,Z)

```

Another example from `surf` will clear the idea more. Create a 2D grid with uniformly spaced x -coordinates and y -coordinates in the interval $[-2, 2]$. Then evaluate and plot the function $f(x, y) = xe^{-x^2-y^2}$ over the 2D grid. It is shown in figure 1.4.

```

1 x = -2:0.25:2;
2 y = x;
3 [X,Y] = meshgrid(x);
4 F = X.*exp(-X.^2-Y.^2);
5 surf(X,Y,F)

```

You can have a title on a graph, label each axis, change the font and font size, set up the scale for each axis and have a legend for the graph. You can also have multiple graphs per page. For example, we will add a title and axis labels to a chart by using the `title`, `xlabel`, and `ylabel` functions, as shown in figure 1.5. We can also add a legend to the graph that identifies each data set using the `legend` function. Beware to specify the legend descriptions in the order that you plot the lines.

```

1 x = linspace(-2*pi,2*pi,100);
2 y1 = sin(x);

```

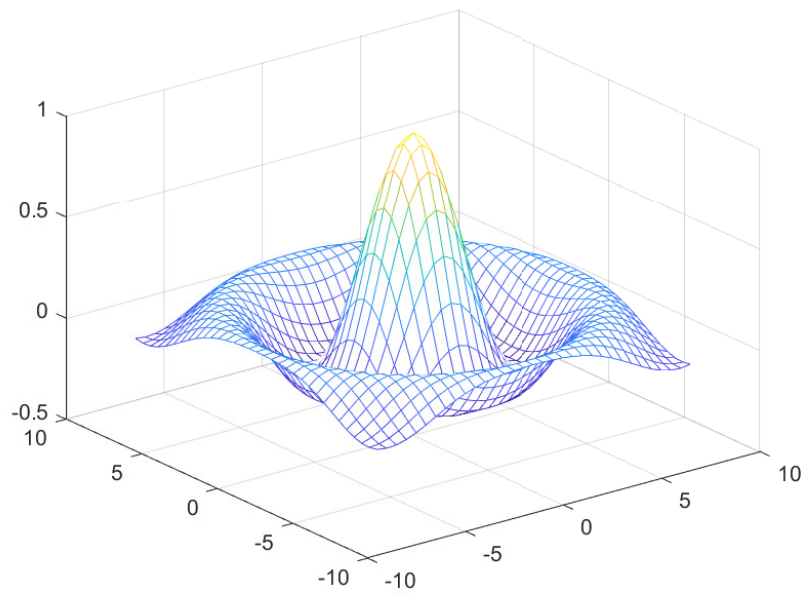



Figure 1.3: A sample mesh plot

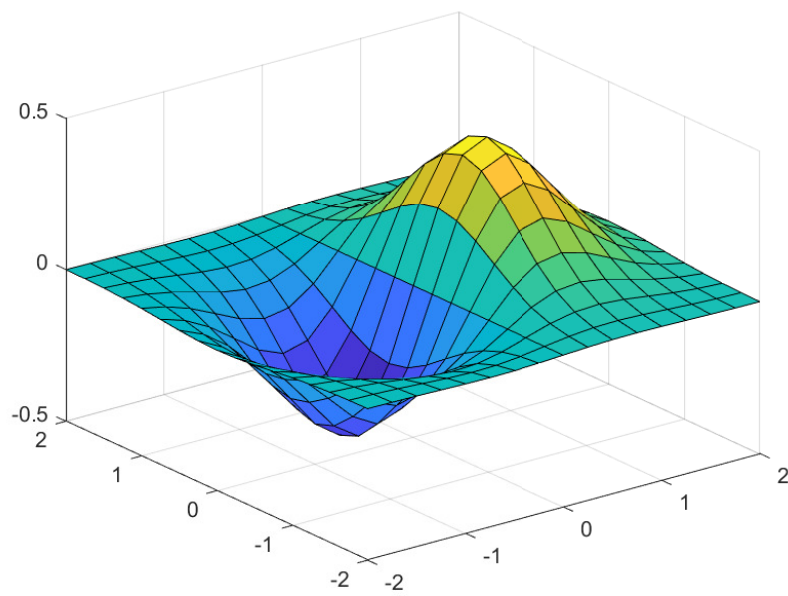


Figure 1.4: A sample surf plot

```

3 | y2 = cos(x);
4 | figure
5 | plot(x,y1,x,y2)
6 | title('Line Plot of Sine and Cosine Between  $-2\pi$  and  $2\pi$ ')

```

```

7 xlabel('-2\pi < x < 2\pi')
8 ylabel('Sine and Cosine Values')
9 legend({'y = sin(x)', 'y = cos(x)'})

```

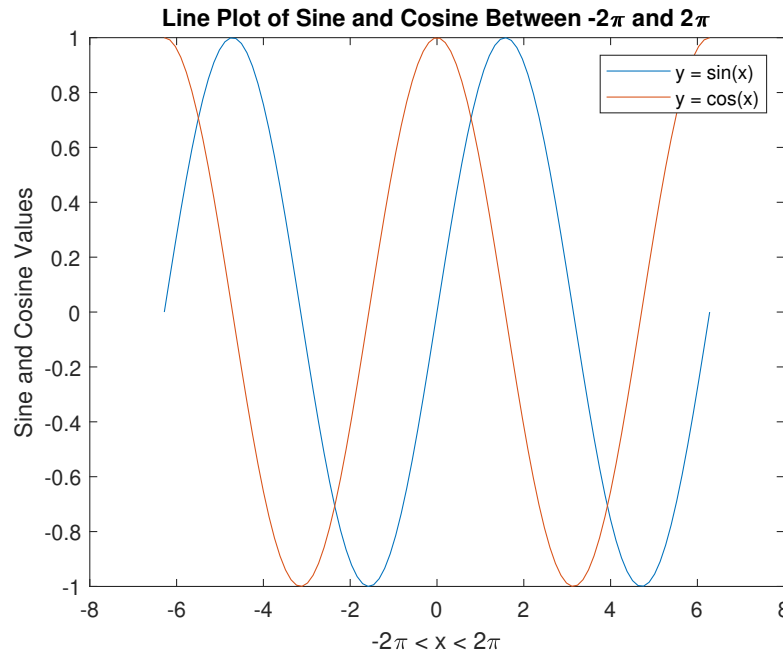


Figure 1.5: A sample plot with title and axes labeling

You can use the `subplot` command to obtain several smaller “subplots” in the same figure. The syntax is `subplot(m,n,p)`. This command divides the Figure window into an array of rectangular panes with m rows and n columns. The variable p tells MATLAB to place the output of the plot command following the `subplot` command into the p th pane. For example, `subplot(3,2,5)` creates an array of six panes, three panes deep and two panes across, and directs the next plot to appear in the p th pane (in the bottom left corner), as shown in figure 1.6. Also `xlim(limits)` sets the x -axis limits for the current axes or chart. Specify limits as a two-element vector of the form `[xmin xmax]`, where `xmax` is greater than `xmin`. Or `axis` can also be used to specify axes limit.

```

1 x = 0:0.01:5;
2 y = exp(-1.2*x).*sin(10*x+5);
3 subplot(1,2,1)
4 plot(x,y)
5 xlabel('x')
6 ylabel('y')
7 xlim([0 5])
8 ylim([-1 1])
9

```

```

10 x = -6:0.01:6;
11 y = abs(x.^3-100);
12 subplot(1,2,2)
13 plot(x,y)
14 xlabel('x')
15 ylabel('y')
16 axis([-6 6 0 350])

```

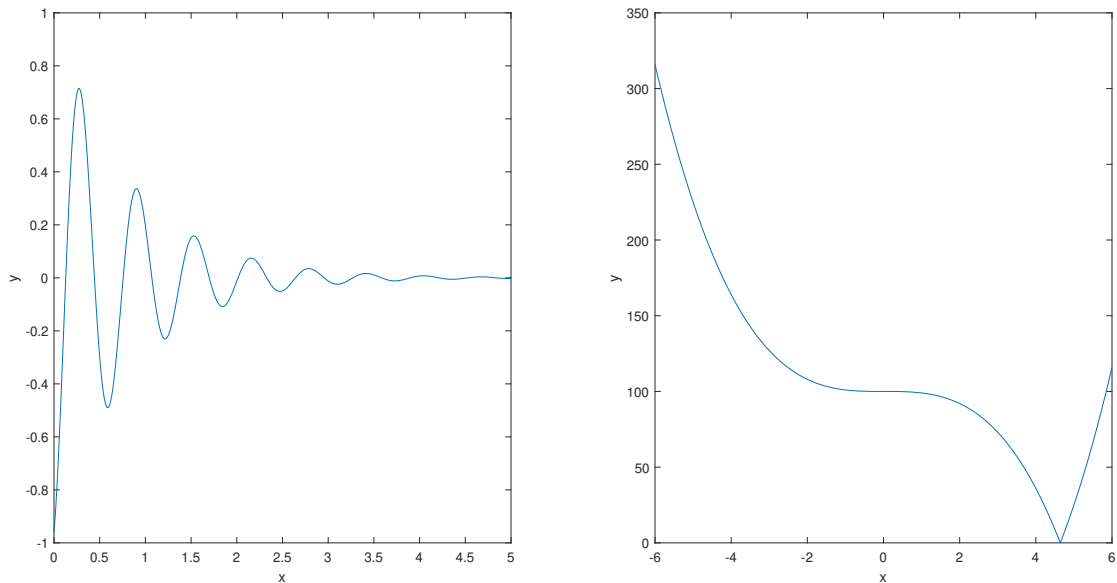


Figure 1.6: A sample subplot

1.1.4 Element-Wise Operations

Sometimes we need to carry out operations on individual elements of an array.

```

1 % Let us start with an array x
2 x = [0, 0.25, 0.50, 0.75, 1.00];
3
4 % We would like to square each element of the array. In matlab, this can be
  % done with:
5 % x.^2 % Notice the dot (.) in front of exponentiation (^).
6
7 x2 = x.^2
8 x2 =
9      0      0.0625      0.2500      0.5625      1.0000

```

³Most of the content of this section is taken from the book *A Concise Introduction to Matlab 3rd ed.* by William J.

Similarly, all algebraic operations can be carried out element–wise on arrays and matrices.

1.2 Vectors, Arrays and Matrices

One of the strengths of MATLAB is its ability to handle collections of numbers, called arrays, as if they were a single variable³. A numerical *array* is an ordered collection of numbers (a set of numbers arranged in a specific order). An example of an array variable is one that contains the numbers 0, 4, 3, and 6, in that order. We use square brackets to denote that the variable x contain this collection by typing $x = [0, 4, 3, 6]$. The elements of the array may also be separated by spaces, but commas are preferred to improve readability and avoid mistakes.

1.2.1 Vector Algebra

```
1 %% Representation of Vectors
2
3 % List of numbers
4 % Enclose in square brackets
5 % Matlab treat every number as a vector
6
7 a = [1 2 3 4 5]
8 b = [6 7 8 9 10]
9
10 %% Semicolon ';' The semicolon can be used to construct arrays,
11 % suppress output from a MATLAB command,
12 % or to separate commands entered on the same line.
13
14 % Addition of vectors
15 add = a + b
16
17 % Subtraction of vectors
18 sub = b - a
19
20 %% Multiplication of a vector by a scalar
21
22 g = 5*a
23 h = 4*b
24
25 % You can also check the following commands as well.
26 %(a) a + b
27 %(b) a* b
28 %(c) a*c
29 %(d) a.*d
```

```

30 %(e) a.*b
31
32 %% norm can be used to find the magnitude of a vector
33
34 aNorm = norm(a)
35 bNorm = norm(b)
36
37 %% Dot Product of Real Vectors
38
39 A1 = [4 -1 2];
40 B1 = [2 -2 -1];
41
42 DotProduct = dot(A1,B1)
43
44 %% Cross Product of Real Vectors
45
46 a1 = [1 2 3];
47 b1 = [4 5 6];
48
49 crossproduct = cross(a1,b1)
50
51 %% Angle theta
52
53 u = [1 2 0];
54 v = [1 0 0];
55
56 CosTheta = dot(u,v)/(norm(u)*norm(v))
57
58 %% theta in Degrees
59
60 ThetaInDegree = acosd(CosTheta)

```

And their outputs are shown below.

```

1 a =
2     1     2     3     4     5
3 b =
4     6     7     8     9    10
5 add =
6     7     9    11    13    15
7 sub =
8     5     5     5     5     5
9 g =
10     5    10    15    20    25
11 h =
12    24    28    32    36    40

```

```

13 aNorm =
14     7.4162
15 bNorm =
16    18.1659
17 DotProduct =
18     8
19 crossproduct =
20    -3     6    -3
21 CosTheta =
22     0.4472
23 ThetaInDegree =
24    63.4349

```

Arrays can be combined to create matrices. To create a matrix that has multiple rows, separate the rows with semicolons. To transpose a matrix, use a single quote ('). The matrix operators for multiplication, division, and power each have a corresponding array operator that operates element-wise.

```

1  >> a = [1 2 3; 4 5 6; 7 8 10]
2  a =
3      1     2     3
4      4     5     6
5      7     8    10
6  >> aTrans = a'
7  aTrans =
8      1     4     7
9      2     5     8
10     3     6    10
11 >> a3 = a.^3
12 a3 =
13      1     8    27
14     64    125   216
15    343    512  1000

```

Concatenation is the process of joining arrays to make larger ones. In fact, you made your first array by concatenating its individual elements. The pair of square brackets [] is the concatenation operator.

```

1  >> A = [a,a]
2  A =
3      1     2     3     1     2     3
4      4     5     6     4     5     6
5      7     8    10     7     8    10

```

Concatenating arrays next to one another using commas is called horizontal concatenation. Each array must have the same number of rows. Similarly, when the arrays have the same number

of columns, you can concatenate vertically using semicolons.

```
1 >> A = [a; a]
2 A =
3     1     2     3
4     4     5     6
5     7     8    10
6     1     2     3
7     4     5     6
8     7     8    10
```

1.3 Loops

To use MATLAB to solve many physics problems you have to know how to write loops⁴. A loop is a way of repeatedly executing a section of code. It is so important to know how to write them that several common examples of how they are used will be given here. The two kinds of loops we will use are the `for` loop and the `while` loop.

1.3.1 For Loop

The `for` loop looks like this:

```
for n = 1:N . . . end
```

which tells MATLAB to start n at 1, then increment it by 1 over and over until it counts up to N , executing the code between `for` and `end` for each new value of n . For example, let's find the sum

of the series $\sum_{n=1}^N \frac{1}{n^2}$.

```
1 s = 0;           % set a variable to zero so that 1/n^2 can be repeatedly added
   to it
2 N = 10000;       % set the upper limit of the sum
3
4 for n=1:N        % start of the loop
5     s = s + 1/n^2; % add 1/n^2 to s each time, then put the answer back into
   s
6 end              % end of the loop
7
8 fprintf('Sum = %g \n',s) % print the answer
```

And the Sum comes out to be 1.64483.

⁴Most of the content of this section is taken from the book *A Concise Introduction to Matlab 3rd ed.* by William J. Palm III

1.3.2 If Else Condition

`if expression, statements, end` evaluates an expression, and executes a group of statements when the expression is true. An expression is true when its result is nonempty and contains only nonzero elements (logical or real numeric). Otherwise, the expression is false. The `elseif` and `else` blocks are optional. The statements execute only if previous expressions in the `if . . . end` block are false. An `if` block can include multiple `elseif` blocks.

For example, the value of $f(x)$ is $-3x$ when $x < 0$; $x(x - 3)$ when x is in $[0, 2]$ and $\log(x - 3)$ otherwise. To calculate $f(x)$, a simple MATLAB program can be written as:

```
1 if x < 0
2     f = -3*x
3 elseif x <= 2
4     f = x*(x-3)
5 else
6     f = log10(x-1)
7 end
```


Chapter 2

Physics with MATLAB

2.1 Mechanics

2.1.1 Gravitational Force between Two Masses

```
1 % This program calculates and displays the Gravitational Force between two
  masses
2
3 % Input Variables
4 mass_1 = input('Enter the value of mass 1 (in kilogram) ');
5 mass_2 = input('Enter the value of mass 2 (in kilogram)');
6 distance = input('Enter the value of distance ');
7 G = 6.67*10^(-11); % Gravitational constant in the units of Nm^2/kg^2:
8
9 % Calculation
10 force = (G .* mass_1 .* mass_2)./(distance.^2); % computes the value of
  Gravitational Force
11 display(['Force between the masses is = ',num2str(force),' newtons.'])
```

And the outputs (from command window) with arbitrary masses and distance are:

```
1 Enter the value of mass 1 (in kilogram) 2.5
2 Enter the value of mass 2 (in kilogram)1.093
3 Enter the value of distance 1.0e-5
4 Force between the masses is = 1.8226 newtons.
```

2.1.2 Gravitational Force—An Inverse Square Law

```
1 % We will show that gravitational force follows inverse square law.
2
3 % Input Variables
```

```

4 mass_1 = input('Enter the value of mass 1 (in kilogram) = ');
5 mass_2 = input('Enter the value of mass 2 (in kilogram) = ');
6 r = input('Enter a positive value of distance (in meters) = ');
7 G = 6.67*10^(-11); % Gravitational constant in the units of Nm^2/kg^2
8
9 % Calculations
10 dr = r/200; % calculates the Step Size
11 distance = r:dr:2*r; % creates an array of 200 values
12 force = (G .* mass_1 .* mass_2)./(distance.^2);
13 plot(distance.^2,force,'LineWidth',1.5)
14 xlabel('Distance in meters^2')
15 ylabel('Force in kgm/s^2')

```

And the outputs (from command window) with arbitrary masses and distance and the behavior of force with distance is shown below.

```

1 Enter the value of mass 1 (in kilogram) = 2.5
2 Enter the value of mass 2 (in kilogram) = 1.093
3 Enter a positive value of distance (in meters) = 1.0e-9

```

2.1.3 Free Fall Motion

```

1 % Input Variable:
2 % tfinal = final time (in seconds)
3 %
4 % Output Variables:
5 % t = array of times at which speed is % computed (in seconds)
6 % v = array of speeds (meters/second)
7
8 g = 9.81; % Acceleration in SI units
9 tfinal = input('Enter final time (in seconds): ');
10 dt = tfinal/500;
11 t = 0:dt:tfinal; % Creates an array of 501 time values
12 v = g*t;
13 plot(t,v)
14 xlabel('t in sec')
15 ylabel('v in m/s')

```

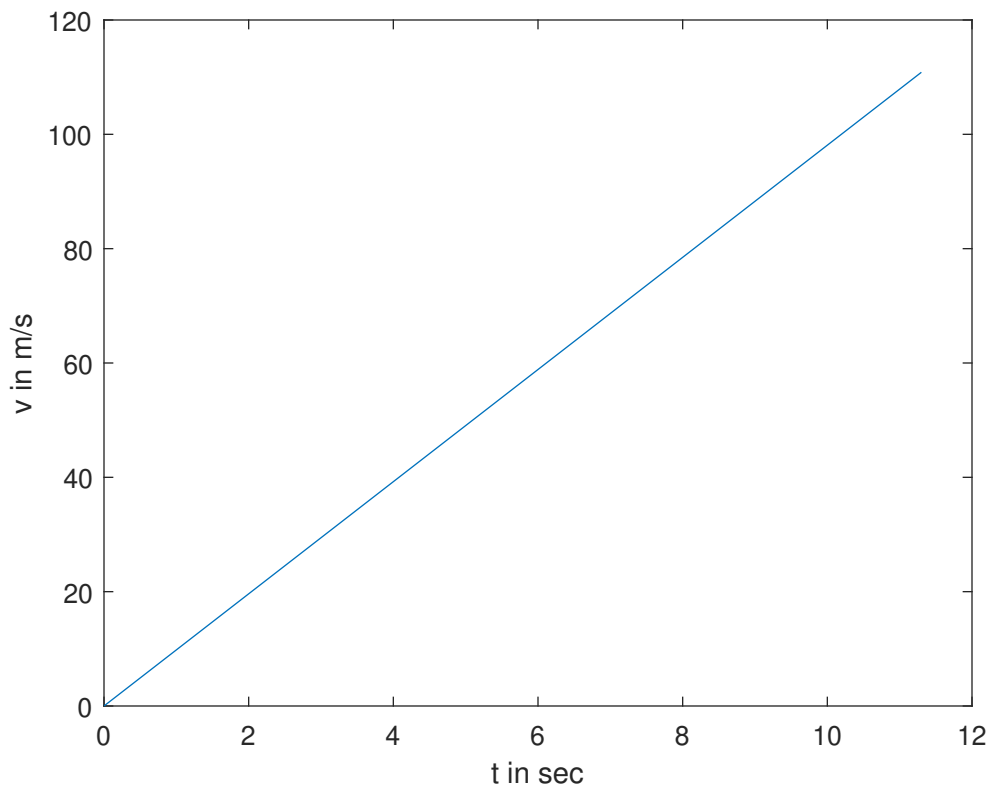


Figure 2.1: Falling objects in free fall (for $t = 11.3$ seconds)

2.1.4 Projectile Motion without making Custom Functions

```

1 %% Projectile's trajectory
2
3 x = 1:0.1:80;
4 g = 9.8;
5 v0 = 30;
6 theta = 30;
7 y = x .* tand(theta) - (x.^2 * g)/(2 * v0^2 * (cosd(theta)^2));
8 plot(x,y,'r','linewidth',1.5)
9
10 hold on
11 theta1 = 45;
12 y1 = x .* tand(theta1) - (x.^2 * g)/(2 * v0^2 * (cosd(theta1)^2));
13 plot(x,y1,'b','linewidth',1.5)
14
15 theta2 = 60;
16 y2 = x .* tand(theta2) - (x.^2 * g)/(2 * v0^2 * (cosd(theta2)^2));
17 plot(x,y2,'g','linewidth',1.5)
18

```

```

19 xlabel('x')
20 ylabel('y')
21 legend('\theta = 30^{o}', '\theta = 45^{o}', '\theta = 60^{o}')
22 hold off
23
24 %% Range of projectile
25
26 theta_new = 0:0.01:90;
27 R = (v0^2 * sind(2*theta_new))/g;
28 plot(theta_new,R,'r','linewidth',1.5)
29 xlabel('Angle (in degrees)')
30 ylabel('Range (in meters)')
31
32 %% Height of projectile
33
34 theta_new = 0:0.1:90;
35 H = (v0^2 * sind(theta_new).^2)/(2*g);
36 plot(theta_new,H,'r','linewidth',1.5)
37 xlabel('Angle (in degrees)')
38 ylabel('Height (in meters)')

```

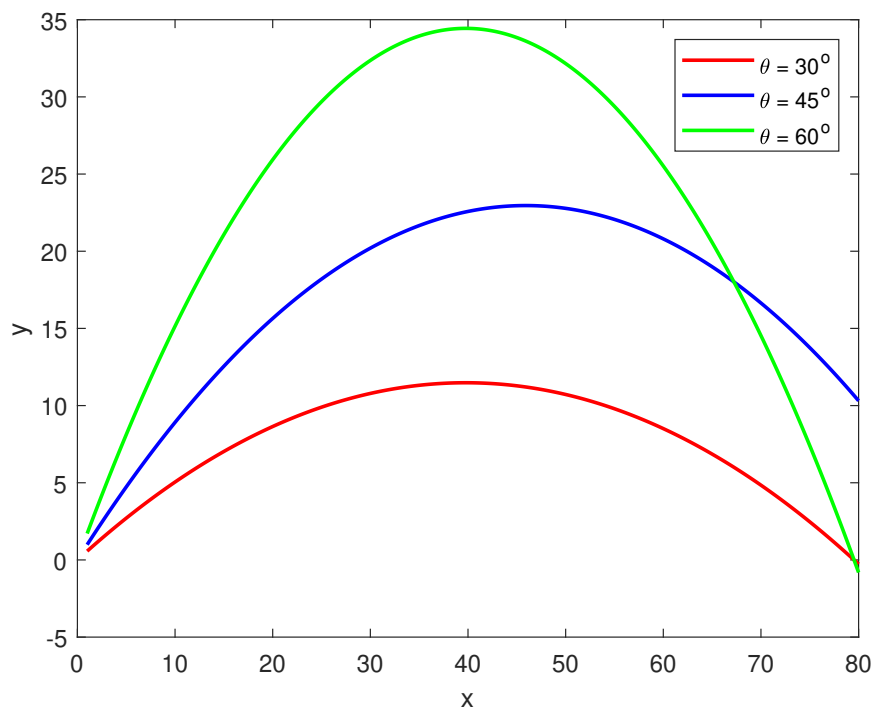


Figure 2.2: Trajectory of a projectile

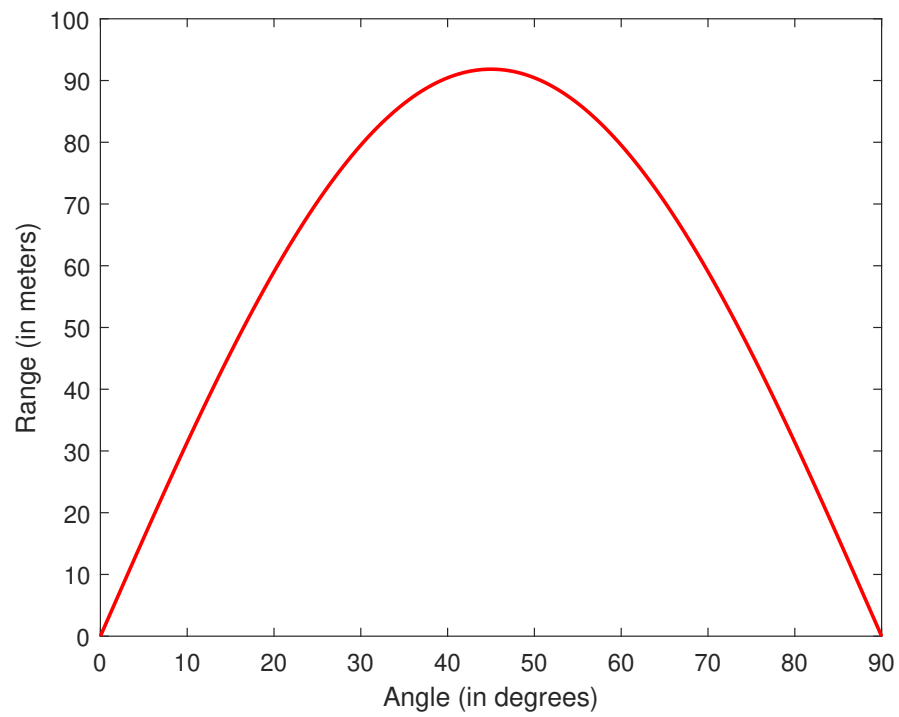


Figure 2.3: Range of a projectile

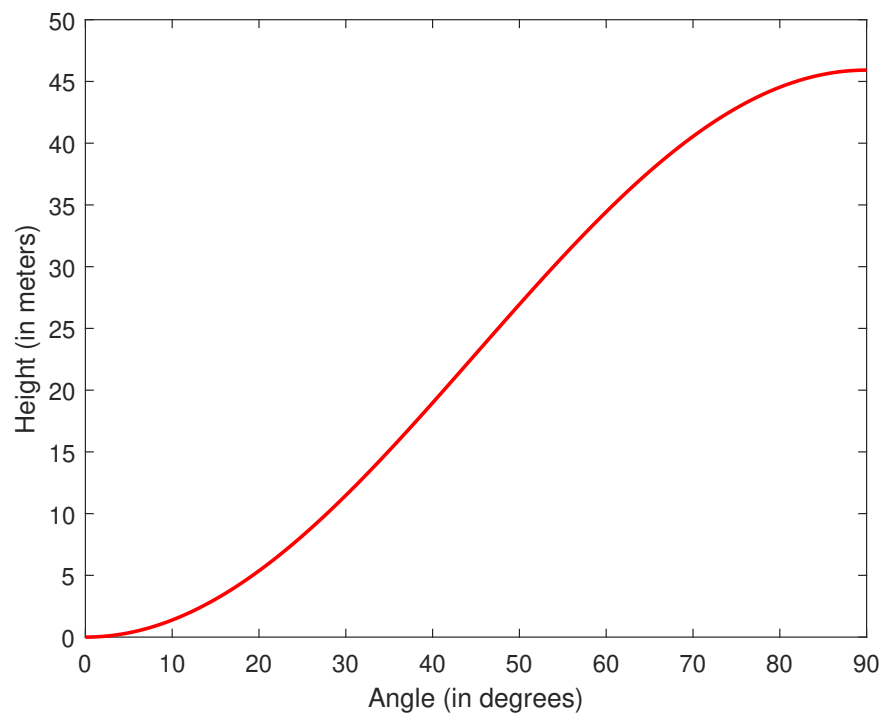


Figure 2.4: Height of a projectile

2.1.5 Projectile Motion by making Custom Functions

To run any script, all files should be in the same directory.

- Make a custom function for the height of the projectile.

```
1 %% Syntax
2 % Height = HProjectile(Angle of Incidence in degrees,velocity in m/s)
3 % File    = HProjectile.m
4
5 %% Function
6
7 function H = HProjectile(Theta,v)
8 H = (v.^2).*((sind(Theta)).^2)./(2.*(9.8));
9 end
```

- Make a custom function for the range of the projectile.

```
1 %% Syntax
2 % Range = RProjectile(Angle of Incidence in degrees,velocity in m/s)
3 % File    = RProjectile.m
4
5 %% Function
6
7 function R = RProjectile(Theta,v)
8 R = (v.^2).*(sind(2.*Theta))./(9.8);
9 end
```

- Make a custom function for the trajectory of the projectile.

```
1 %% Syntax
2 % trProjectile(Angle of Incidence in degrees,velocity in m/s)
3 % File    = trProjectile.m
4
5 %% Function
6 function y = trProjectile(Theta,v)
7 x = 0:0.1:RProjectile(Theta,v);
8
9 for i = 1:length(x)
10     y(i) = tand(Theta).*x(i) - ((9.8).*x(i).^2)./(2.*(v.*cosd(Theta)).^2);
11 end
```

```
12 |
13 | plot(x,y,'linewidth',1.5);
14 | grid on
15 | xlabel('Range (m)')
16 | ylabel('Altitude (m)')
```

Now define θ and v in the *command window* or in an a *new MATLAB file* and run `HProjectile(θ ,v)`, `RProjectile(θ ,v)`, `trProjectile(θ ,v)` for output. An example (only values) from command window is shown below.

```
1 >> Theta = 45;
2 >> v = 10;
3 >> H = HProjectile(Theta,v)
4 H =
5     2.5510
6 >> R = RProjectile(Theta,v)
7 R =
8     10.2041
9 >> Y = trProjectile(Theta,v);
```

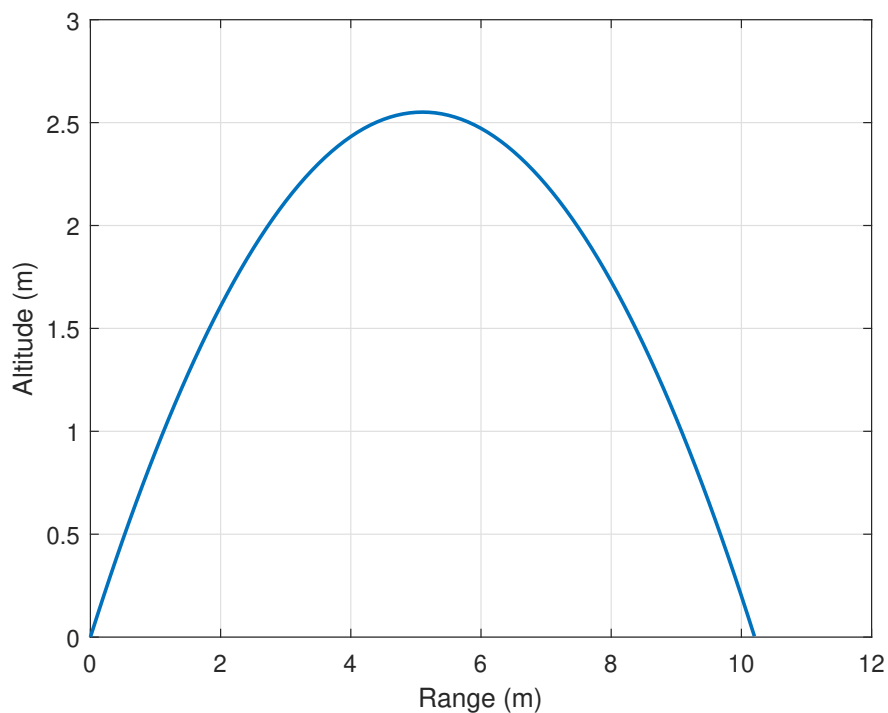


Figure 2.5: Trajectory of a projectile (using custom functions)

2.2 Waves and Oscillations

2.2.1 SHM as Circular Motion

```
1 %% This code will demonstrate from equations of motion of SHM that SHM is a
  type of circular motion.
2 % Code starts from here.
3
4 clear all          % Clears the workspace
5 close all         % Close all previous plots and figures
6 clc              % Clears the command window
7
8
9 N = 1080;         % Declare total phase of oscillation
10 A0 = 10;         % Declare amplitude
11 theta(1) = 0;    % Initial value of theta
12
13
14 for i=1:N;        % For loop to find phase angle
15 % theta(i) is the previous angle and theta(i+1) in the new angle.
16 % theta is measured in radians.
17     theta(i+1) = theta(i) + 1;
18 % Convert theta(i+1) into degrees by multiplying it with (pi/180).
19 % alpha(i) is the new theta(i+1) measured in degrees.
20     alpha(i) = (pi/180) * theta(i+1);
21 end
22
23
24 % In the next three for loops, x represents first simple harmonic
25 % oscillator and y represents second simple harmonic oscillator
26
27
28 % For loop to find position of both SHOs
29 for i=1:N;
30     x(i) = A0 * cos(alpha(i));    % Position equation of first SHO
31     y(i) = A0 * sin(alpha(i));    % Position equation of second SHO
32 end
33
34
35 % For loop to find velocity of both SHOs
36 for i=1:N;
37     vx(i) = - A0 * sin(alpha(i)); % Velocity equation of first SHO
38     vy(i) = A0 * cos(alpha(i));   % Velocity equation of second SHO
39 end
```



```

40
41
42 % For loop to find acceleration of both SHOs
43 for i=1:N;
44     ax(i) = - A0 * cos(alpha(i)); % Acceleration equation of first SH0
45     ay(i) = - A0 * sin(alpha(i)); % Acceleration equation of second SH0
46 end
47
48
49 % Let's plot Phase Angle vs. Position
50 figure(1);
51 hold on % hold ON sets the NextPlot property of the current figure and
    axes to add.
52 plot(alpha,x,'linewidth',1.5) % LineWidth sets the width of line.
53 plot(alpha,y,'linewidth',1.5)
54 xlabel('Phase Angle') % xlabel('text') adds text beside the X-axis.
55 ylabel('Amplitude') % ylabel('text') adds text beside the Y-axis.
56 ylim([-12 12]) % ylim([YMIN YMAX] sets the y limits.
57 legend('x(i)','y(i)')
58 title('Phase Angle vs. Position (in SHM)')
59 grid on % grid ON adds major grid lines.
60 hold off % hold OFF sets the NextPlot property of the current figure and
    axes to replace.
61
62
63 % Let's plot Phase Angle vs. Velocity
64 figure(2);
65 hold on
66 plot(alpha,vx,'linewidth',1.5)
67 plot(alpha,vy,'linewidth',1.5)
68 xlabel('Phase Angle')
69 ylabel('Velocity')
70 ylim([-12 12])
71 legend('v_{x}(i)','v_{y}(i)')
72 title('Phase Angle vs. Velocity (in SHM)')
73 grid on
74 hold off
75
76
77 % Let's plot Phase Angle vs. Acceleration
78 figure(3);
79 hold on
80 plot(alpha,ax,'linewidth',1.5)
81 plot(alpha,ay,'linewidth',1.5)
82 xlabel('Phase Angle')

```

```

83     ylabel('Acceleration')
84     ylim([-12 12])
85     legend('a_{x}(i)', 'a_{y}(i)')
86     title('Phase Angle vs. Acceleration (in SHM)')
87     grid on
88 hold off
89
90
91 % Let's plot to show that both SHOs depict circular motion
92 figure(4);
93     plot(x,y, 'linewidth',1.5)
94     xlabel('x(t)')
95     ylabel('y(t)')
96     xlim([-12 12])
97     ylim([-12 12])
98     title('Simple Harmonic Motion as Circular Motion')
99     grid on
100
101
102 % Let's combine first three plots into one.
103 figure;
104 subplot(3,1,1)           % Write help
    subplot in Command Window to understand it
105 title('Phase Angle vs. Position, Velocity and Acceleration (in SHM)');
106 hold on
107     plot(alpha,x, 'linewidth',1.5)
108     plot(alpha,y, 'linewidth',1.5)
109     ylabel('Amplitude')
110     ylim([-12 12])
111     legend('x(i)', 'y(i)')
112     grid on
113 hold off
114
115
116 subplot(3,1,2)
117 hold on
118     plot(alpha,vx, 'linewidth',1.5)
119     plot(alpha,vy, 'linewidth',1.5)
120     ylabel('Velocity')
121     ylim([-12 12])
122     legend('v_{x}(i)', 'v_{y}(i)')
123     grid on
124 hold off
125
126

```

```

127 subplot(3,1,3)
128 hold on
129 plot(alpha,ax,'linewidth',1.5)
130 plot(alpha,ay,'linewidth',1.5)
131 xlabel('Phase Angle')
132 ylabel('Acceleration')
133 ylim([-12 12])
134 legend('a_{x}(i)', 'a_{y}(i)')
135 grid on
136 hold off
137
138 %% Task completed.

```

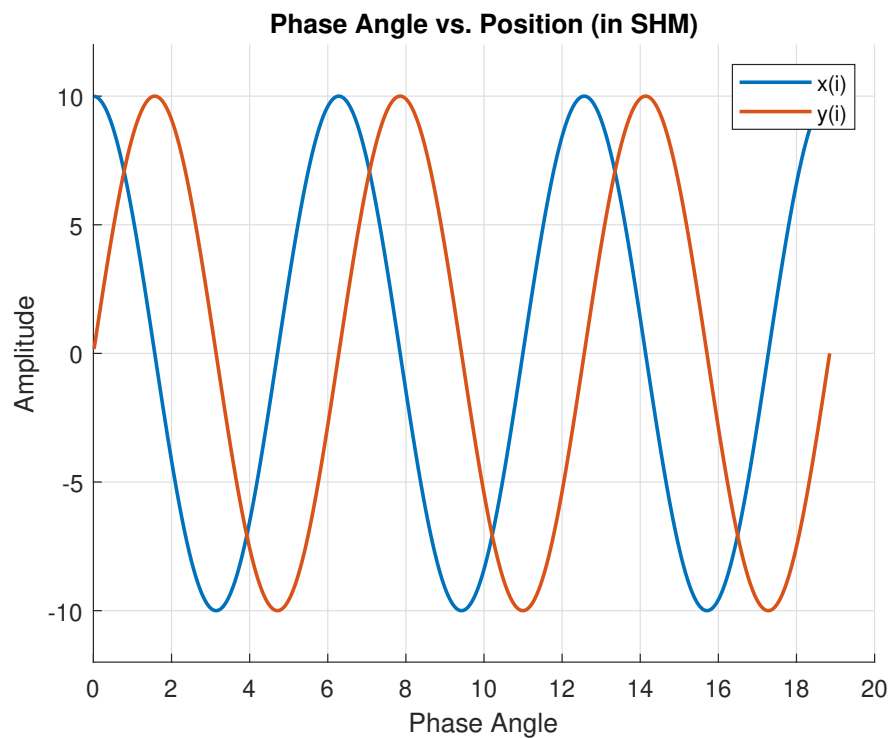


Figure 2.6: Phase Angle vs. Position

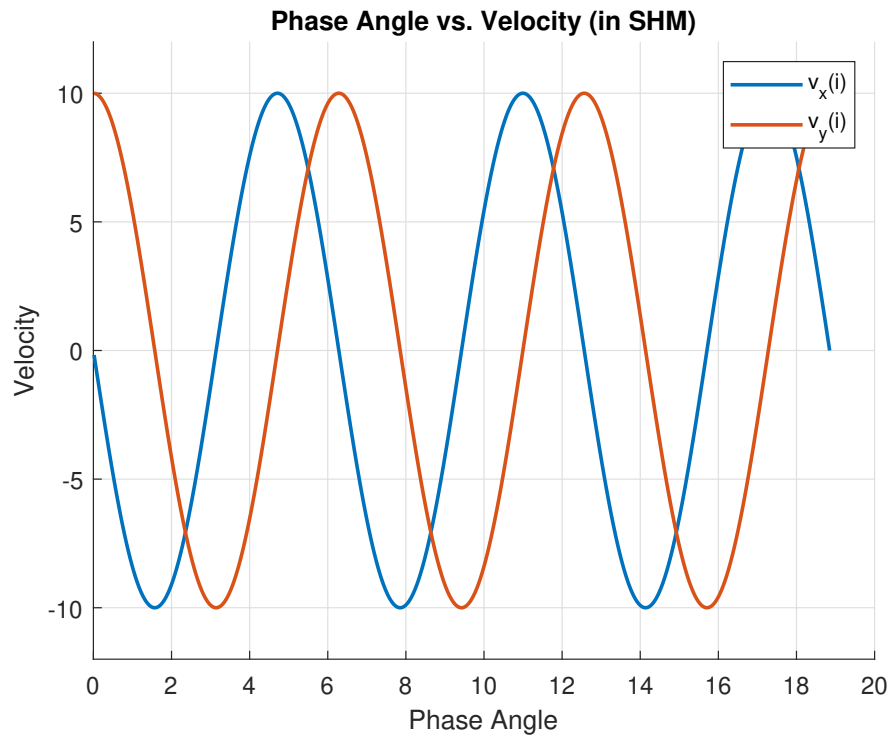


Figure 2.7: Phase Angle vs. Velocity

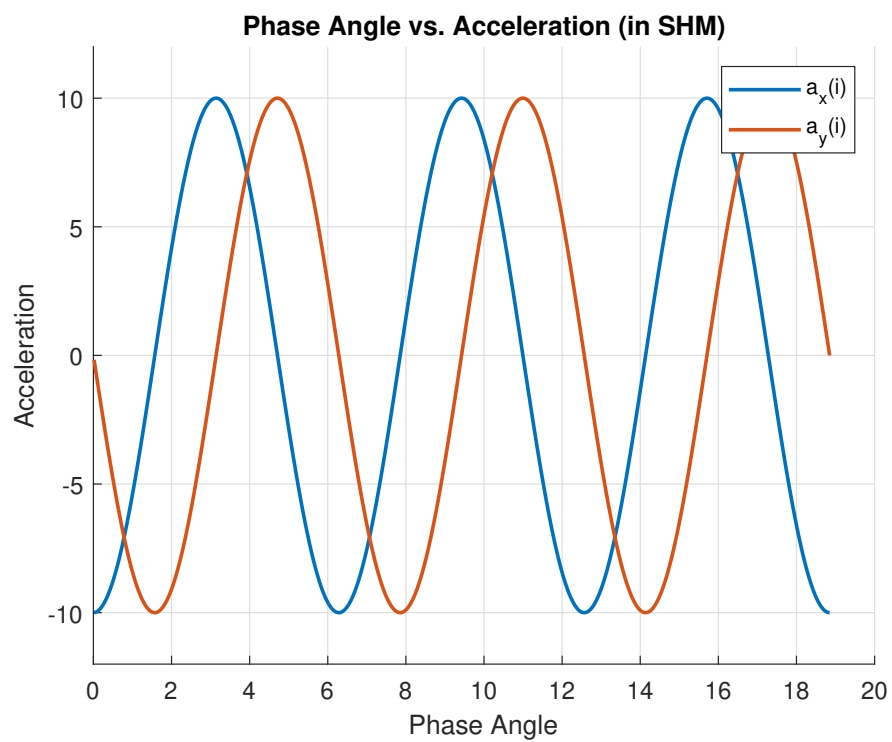


Figure 2.8: Phase Angle vs. Acceleration

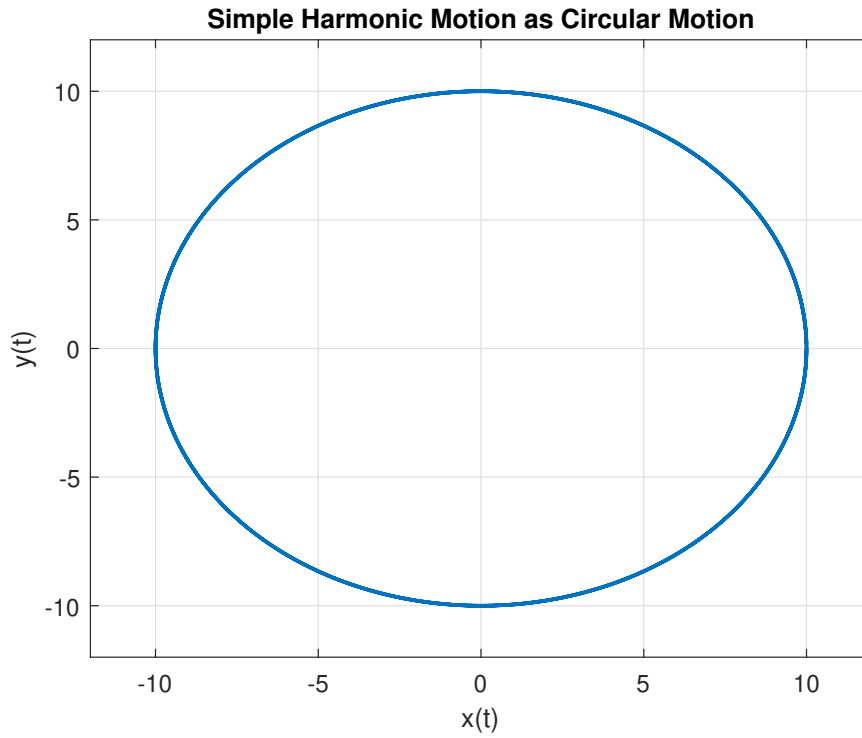


Figure 2.9: SHM as circular motion

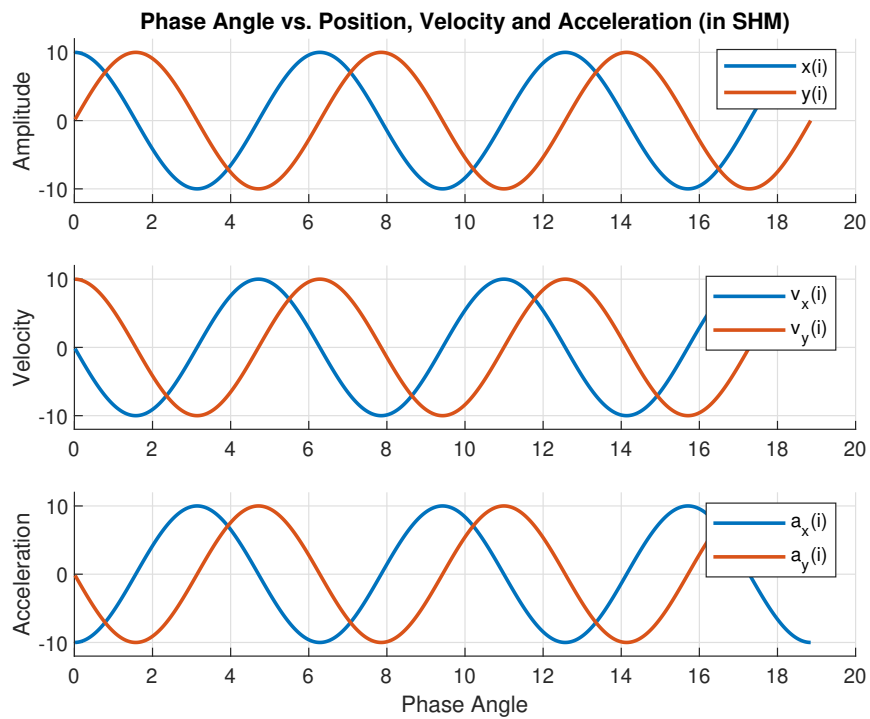


Figure 2.10: Phase Angle vs. Position, Velocity and Acceleration

2.3 Electricity and Magnetism

2.3.1 Coulomb Force between Charges

```
1 %% Coulomb law for two point charges
2
3 eps0 = 8.854e-12;
4 kC = 1/(4*pi*eps0);
5 q1 = -1e-13;
6 q2 = +1e-10;
7 r = [-12:0.1:12].*1e-12;
8
9 F = (kC*q1*q2)./(r.*r);
10
11 plot(r,F,'LineWidth',1.5)
12 xlabel('Distance (in meters)')
13 ylabel('Coulomb force (in newtons)')
```

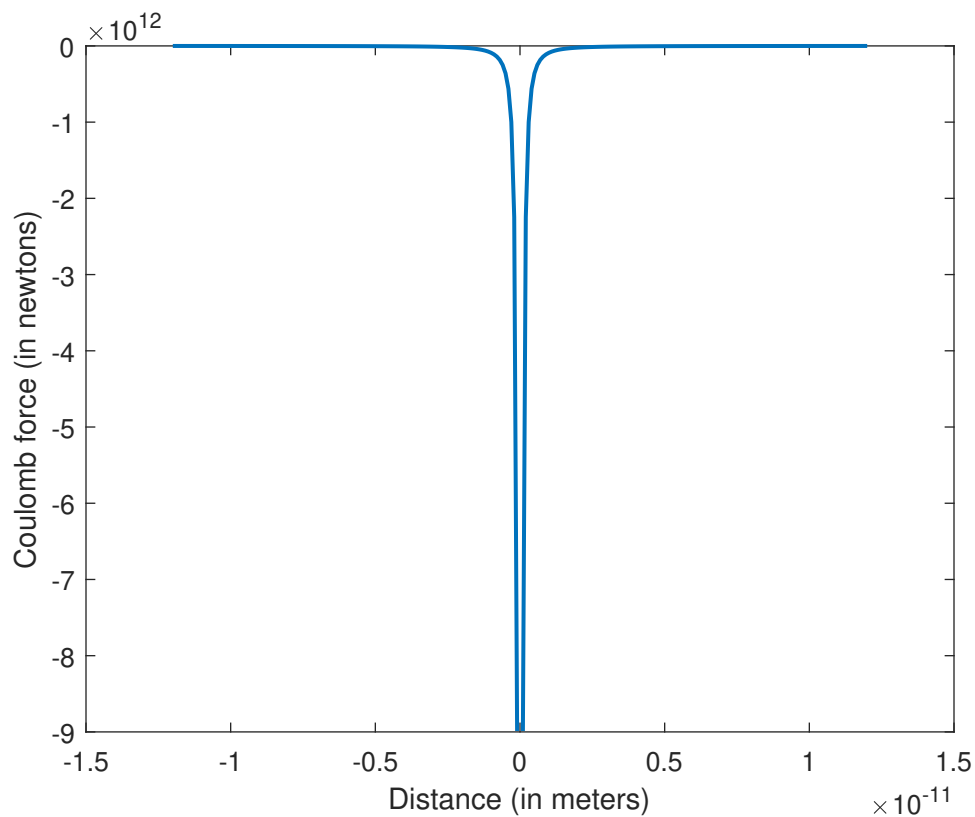


Figure 2.11: Electrostatic (Coulomb) force as inverse square law

2.3.2 Coulomb Force between Charges using For Loop

```
1 %% Cforce – Program to compute Coulomb force between charges
2 % Source: https://ualr.edu/dcwold/phys2322/cforce/cforce.html
3
4 clear all; help Cforce; % Clear memory; print header
5
6 %@ Enter your username
7 fprintf('Enter your username (userid); \n');
8 fprintf('USERNAME: ABName \n');
9 Username = input(' USERNAME: ','s'); % Read input as a text string
10 fprintf('\n');
11
12 %@ Initialize variables (e.g., positions of charges, physical constants)
13 NCharges = input('Enter the number of charges: ');
14 for iCharge=1:NCharges
15     fprintf('—— \n For charge #%g \n',iCharge);
16     r_in = input('Enter position (in m) as [x y]: ');
17     x(iCharge) = r_in(1); % x-component of position
18     y(iCharge) = r_in(2); % y-component of position
19     q(iCharge) = input('Enter charge (in C): ');
20 end
21
22 %@ Find xmin, xmax, ymin, and ymax
23 xmin = min(x)-1;
24 ymin = min(y)-1;
25 xmax = max(x)+1;
26 ymax = max(y)+1;
27
28 Epsilon0 = 8.85e-12; % Permittivity of free space (C^2/(N m^2))
29 Constant = 1/(4*pi*Epsilon0); % Useful constant
30
31 %@ Loop over charges to compute the force on each charge
32 fprintf('\n\n Forces are: \n\n');
33 for iCharge = 1:NCharges
34
35     Fx = 0.0; % Initialize components of total force to zero
36     Fy = 0.0;
37
38     %@ Loop over other charges to compute force on this charge
39     for jCharge = 1:NCharges
40         if( iCharge ~= jCharge ) % If iCharge NOT equal to jCharge
41
42             %@ Compute the components of vector distance between two charges
43             xij = x(iCharge) - x(jCharge);
```

```

44     yij = y(iCharge) - y(jCharge);
45     Rij = sqrt(xij^2 + yij^2);
46
47     %@ Compute the x and y components of the force between
48     %@ these two charges using Coulomb's law
49
50     Fx = Fx + Constant*q(iCharge)*q(jCharge)*xij/Rij^3;
51     Fy = Fy + Constant*q(iCharge)*q(jCharge)*yij/Rij^3;
52
53     end
54 end
55 Fxnet(iCharge) = Fx;
56 Fynet(iCharge) = Fy;
57 %@ Print out the total force on this charge due to the others
58 fprintf('Force on charge #%g is: \n',iCharge);
59 fprintf(' x-component: %g N \n',Fx);
60 fprintf(' y-component: %g N \n',Fy);
61 end
62
63 %@ Plot position of charges
64 clf; % Clear graphics figure window
65 figure; % Bring figure window forward
66 plot(x,y,'bo');
67 axis([xmin xmax ymin ymax]);
68 for j = 1:NCharges
69     text(x(j),y(j),sprintf(' %g',j));
70 end
71 %@ Add force direction to position of charges
72 hold on;
73 quiver(x,y,Fxnet,Fynet,'r'); % Draw arrows for force
74 title([Username,', ',date,', ', 'Cforce: Position of charges and direction
       of forces']);
75 xlabel('x (m)'); ylabel('y (m)');
76 hold off;

```


17120001, 02-Nov-2019, Cforce: Position of charges and direction of forces

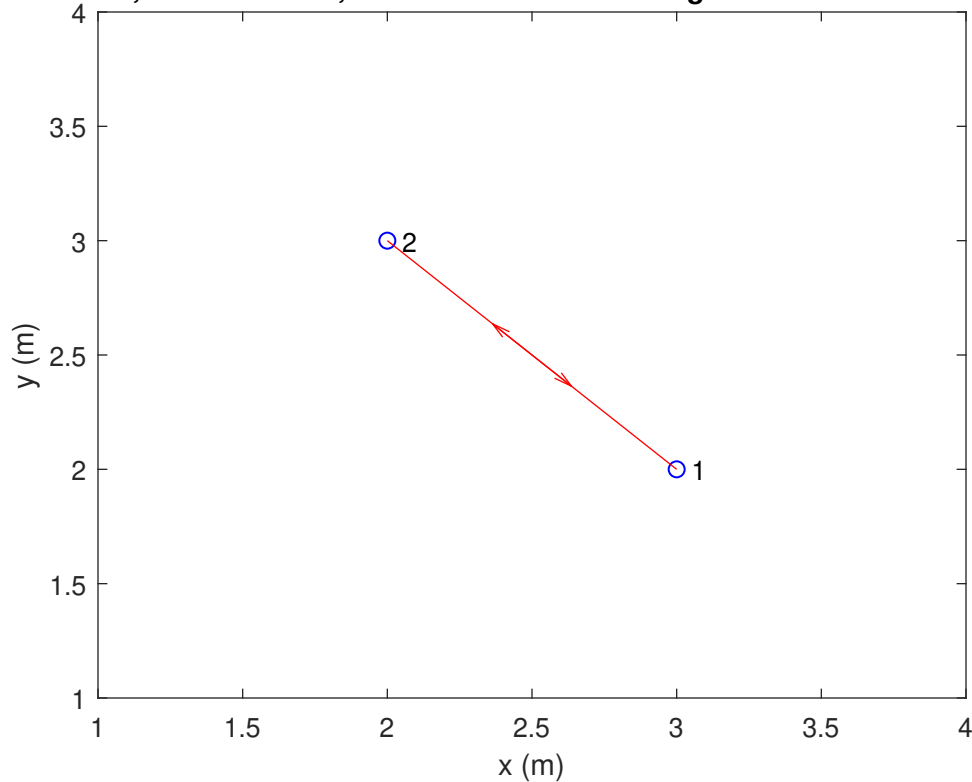


Figure 2.12: Electrostatic (Coulomb) force as inverse square law

2.3.3 Fields due to Discrete and Line Charge Distributions

```

1 %% Fields due to discrete and line charge distributions
2 % Author: S. Mandayam, ECE, Rowan University
3
4 close all;
5 clear;
6
7 [x,y,z] = meshgrid(-1:0.05:1,0.001:0.05:1,-1:0.05:1);
8 % [X,Y,Z] = meshgrid(x,y,z) returns 3-D grid coordinates defined by the
9 % vectors x, y, and z. The grid represented by X, Y, and Z has size
10 % length(y)-by-length(x)-by-length(z).
11
12 % Point Charge
13 E = 1./(x.^2+y.^2+z.^2);
14
15 figure(1);
16 slice(x,y,z,log(E),[-0.9:0.05:0.9],0.9,[-0.9:0.05:0.9]);
17 % slice(X,Y,Z,V,xslice,yslice,zslice) draws slices for the volumetric data
18 % V. Specify X,Y, and Z as the coordinate data. Specify xslice, yslice, and

```

```

19 % zslice as the slice locations using one of these forms:
20 shading interp;
21 % shading interp varies the color in each line segment and face by
22 % interpolating the colormap index or true color value across the line or
23 % face.
24 colormap hsv;
25 % colormap map sets the colormap for the current figure to one of the
26 % predefined colormaps. If you set the colormap for the figure, then axes
27 % and charts in the figure use the same colormap
28 xlabel('x');
29 ylabel('y');
30 zlabel('z');
31 title('Electric Field (Log Magnitude) due to point charge at origin (0,0,0)'
    );
32 axis square;
33 colorbar;
34 % colorbar displays a vertical colorbar to the right of the current axes or
35 % chart. Colorbars display the current colormap and indicate the mapping of
36 % data values into the colormap.
37 rotate3d on;
38 % rotate3d on turns on rotate mode and enables rotation on all axes within
39 % the current figure.
40 pause;
41 % pause temporarily stops MATLAB execution and waits for the user to press
42 % any key.
43
44 % Line Charge
45 E = 1./sqrt(x.^2+y.^2);
46 figure(2);
47 slice(x,y,z,log(E),[-0.9:0.1:0.9],0.9,[-0.9:0.1:0.9]);
48 shading interp;
49 colormap hsv;
50 xlabel('x');
51 ylabel('y');
52 zlabel('z');
53 title('Electric Field (Log Magnitude) due to line charge along z-axis');
54 axis square;
55 colorbar;
56 rotate3d on;

```

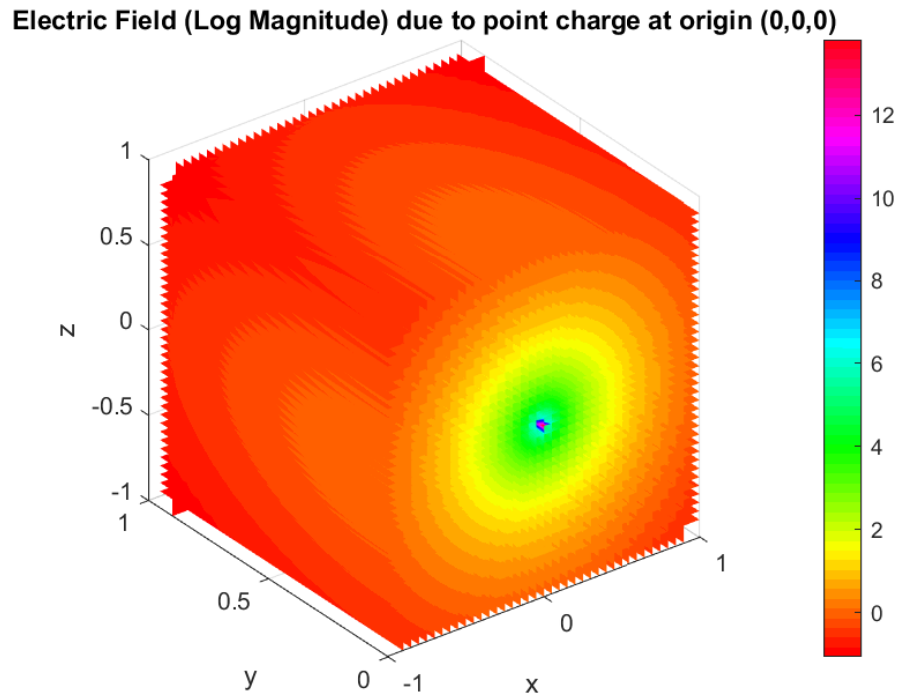


Figure 2.13: Electric Field (Log Magnitude) due to point charge at origin (0,0,0)

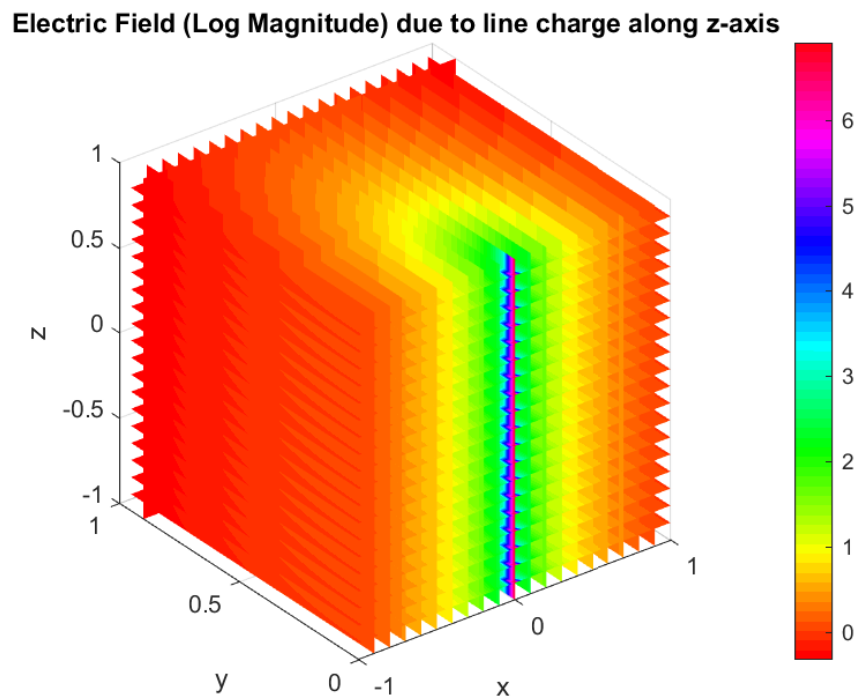


Figure 2.14: Electric Field (Log Magnitude) due to line charge along z-axis

2.3.4 Electric Fields due to dipole in a 2D plane using the Coulomb's Law

```
1 %-----%
2 %       This simple program computes the Electric Fields due to dipole
3 %       in a 2-D plane using the Coulomb's Law
4 %-----%
5
6 %-----%
7 %               REFERENCE
8 % SADIKU, ELEMENTS OF ELECTROMAGNETICS, 4TH EDITION, OXFORD
9 %-----%
10
11
12 clc
13 close all; clear all;
14
15 %-----%
16 %               SYMBOLS USED IN THIS CODE
17 %-----%
18
19 % E = Total electric field
20 % Ex = X-Component of Electric-Field
21 % Ey = Y-Component of Electric-Field
22 % n = Number of charges
23 % Q = All the 'n' charges are stored here
24 % Nx = Number of grid points in X- direction
25 % Ny = Number of grid points in Y-Direction
26 % eps_r = Relative permittivity
27 % r = distance between a selected point and the location of charge
28 % ex = unit vector for x-component electric field
29 % ey = unit vector for y-component electric field
30 %-----%
31
32
33 %-----%
34 %               INITIALIZATION
35 %       Here, all the grid, size, charges, etc. are defined
36 %-----%
37
38 % Constant  $1/(4*\pi*\epsilon_0) = 9*10^9$ 
39 k = 9*10^9;
40
41 % Enter the Relative permittivity
42 eps_r = 1;
43 charge_order = 10^-9; % milli, micro, nano etc..
```

```

44 const = k*charge_order/eps_r;
45
46 % Enter the dimensions
47 Nx = 101; % For 1 meter
48 Ny = 101; % For 1 meter
49
50 % Enter the number of charges.
51 n = 2;
52
53 % Electric fields Initialization
54 E_f = zeros(Nx,Ny);
55 Ex = E_f;
56 Ey = E_f;
57
58 % Vectors initialization
59 ex = E_f;
60 ey = E_f;
61 r = E_f;
62 r_square = E_f;
63
64 % Array of charges
65 Q = [1,-1];
66
67 % Array of locations
68 X = [5,-5];
69 Y = [0,0];
70
71 %-----%
72 %                COMPUTATION OF ELECTRIC FIELDS
73 %-----%
74
75 % Repeat for all the 'n' charges
76 for k = 1:n
77     q = Q(k);
78
79     % Compute the unit vectors
80     for i=1:Nx
81         for j=1:Ny
82
83             r_square(i,j) = (i-51-X(k))^2+(j-51-Y(k))^2;
84             r(i,j) = sqrt(r_square(i,j));
85             ex(i,j) = ex(i,j)+(i-51-X(k))./r(i,j);
86             ey(i,j) = ey(i,j)+(j-51-Y(k))./r(i,j);
87         end
88     end

```

```

89
90
91
92     E_f = E_f + q.*const./r_square;
93
94     Ex = Ex + E_f.*ex.*const;
95     Ey = Ey + E_f.*ey.*const;
96
97 end
98
99 %-----%
100 %                PLOT THE RESULTS
101 %-----%
102
103 x_range = (1:Nx)-51;
104 y_range = (1:Ny)-51;
105 contour_range = -8:0.02:8;
106 contour(x_range,y_range,E_f',contour_range,'linewidth',0.7);
107 axis([-15 15 -15 15]);
108 colorbar('location','eastoutside','fontsize',12);
109 xlabel('x ','fontsize',14);
110 ylabel('y ','fontsize',14);
111 title('Electric field distribution, E (x,y) in V/m','fontsize',14);

```

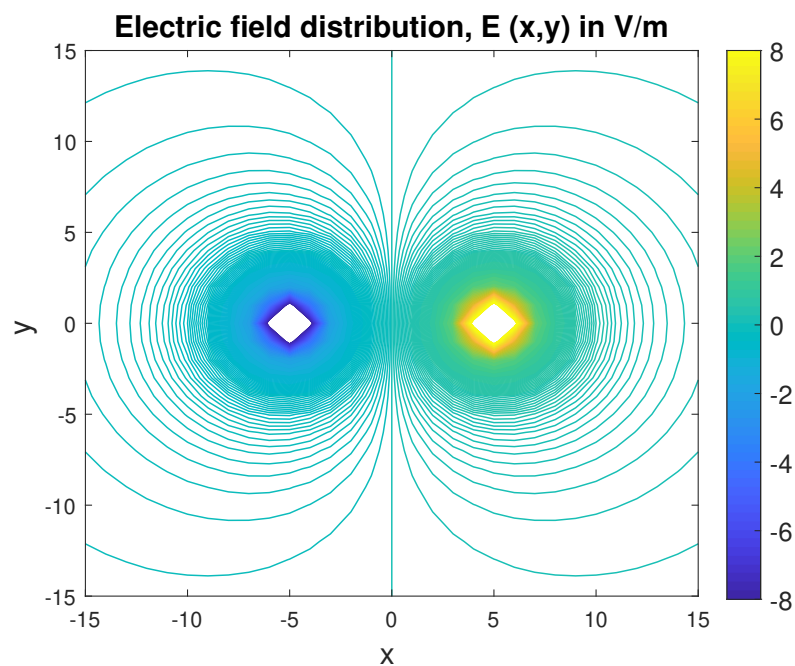


Figure 2.15: Electric field distribution of a dipole

2.3.5 Electric Field due to a Point Charge

```
1 k = 9e9;  
2 r = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20];  
3 E = zeros(20);  
4 q = input('Enter Charge: ');  
5  
6 for i = 1:20  
7     E(i) = (k*q)/(r(i)^2);  
8 end  
9  
10 plot(r,E)  
11 xlabel('Distance (m)')  
12 ylabel('Electric Field (N/C)')
```

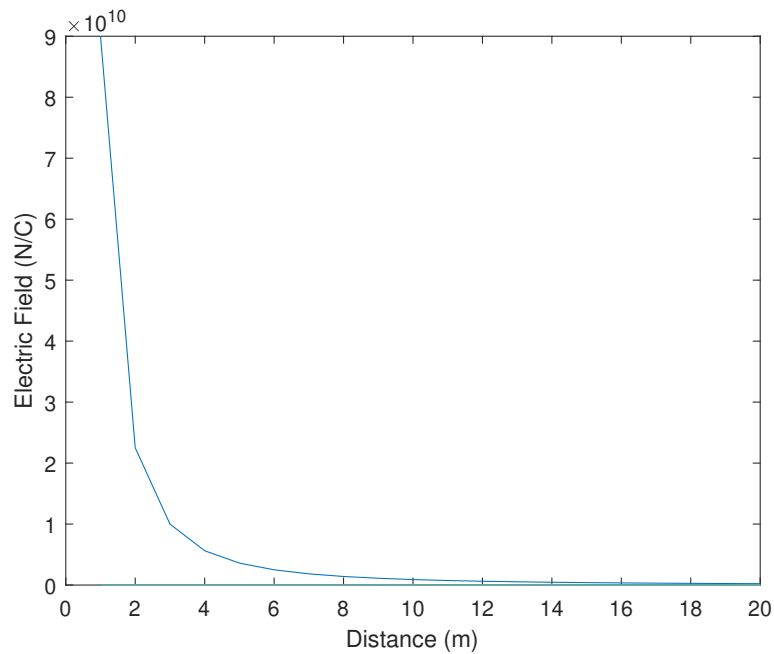


Figure 2.16: Electric Field due to a Point Charge

2.3.6 Electric Field due to a Dipole

```
1 eps0 = 8.85e-12;  
2 k = 1/(2*pi*eps0);  
3 z = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20];  
4 E = zeros(20); % an array of size 20  
5 q = input('Enter Charge: ');  
6 d = input('Enter Distance: ');
```

```

7
8 for i = 1:20
9     E(i) = (k*q*d)/(z(i)^3);
10 end
11
12 plot(z,E)
13 xlabel('Distance (m)')
14 ylabel('Electric Field (N/C)')

```

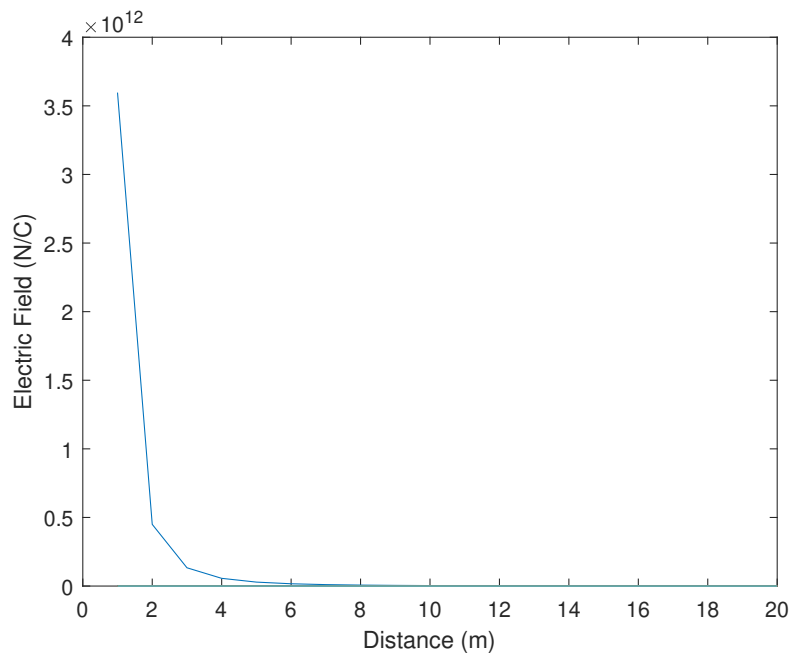


Figure 2.17: Electric Field due to a Dipole

2.3.7 Electric Potential due to a Point Charge

```

1 k = 9e9;
2 r = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20];
3 V = zeros(20);
4 q = input('Enter Charge: ');
5
6 for i = 1:20
7     V(i) = (k*q)/(r(i));
8 end
9
10 plot(r,V)
11 xlabel('Distance (m)')
12 ylabel('Electric Potential (V)')

```

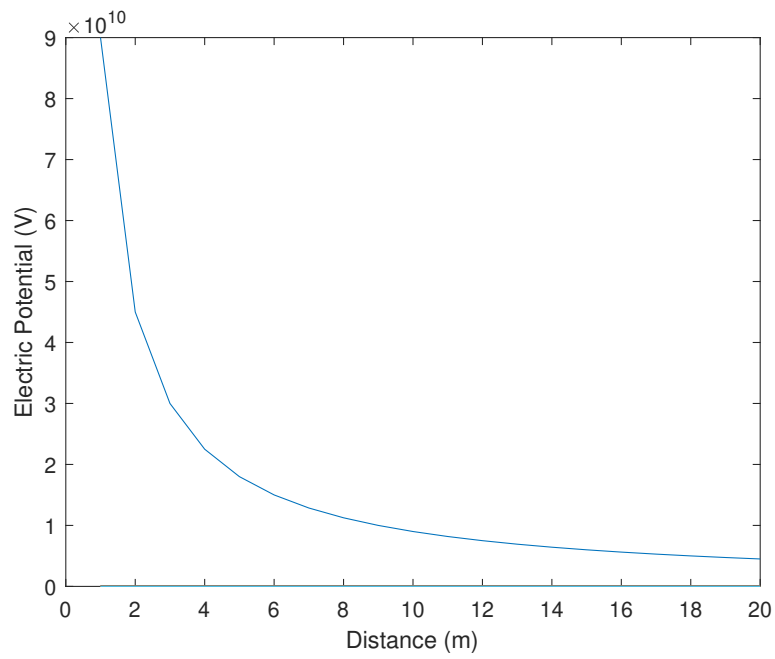



Figure 2.18: Electric Potential due to a Point Charge

Chapter 3

Some Structured Questions with Numerical Analysis

3.1 Questions from Vector Algebra

- Vectors \vec{A} and \vec{B} lie in an xy plane. \vec{A} has magnitude 8.00 and angle 130° ; \vec{B} has components $B_x = -7.72$ and $B_y = -9.20$. What are the angles between the negative direction of the y axis and (a) the direction of A , (b) the direction of the product $\vec{A} \times \vec{B}$, and (c) the direction of $\vec{A} \times (\vec{B} + 3.00\hat{k})$?

```
1 %% Reference: Question 4–47 from Fundamentals of Physics 10th Extended c2014
  ed. by Halliday, Resnick and Walker
2
3 clear all
4 close all
5 clc
6
7 A = 8.0;
8 theta_A = 130;
9
10 vec_A = [A*cosd(theta_A) A*sind(theta_A) 0]
11
12 vec_B = [-7.72 -9.20 0];
13 vec_Y = [0 -1 0];
14
15 mag_A = sqrt(sum(vec_A.*vec_A))
16 mag_Y = sqrt(sum(vec_Y.*vec_Y))
17
18 A_dot_Y = sum(vec_A.*vec_Y)
19 theta_AY = 270 - acosd(A_dot_Y / (mag_A * mag_Y))
20
```

```

21 vec_Bnew = vec_B + [0 0 3];
22 cross_ABnew = cross(vec_A,vec_Bnew)
23 Y_dot_crossABnew = sum(vec_Y.*cross_ABnew)
24 mag_ABnew = sqrt(sum(cross_ABnew.*cross_ABnew))
25 theta_ABY = acosd(Y_dot_crossABnew / (mag_ABnew * mag_Y))

```

The outputs are as follows.

```

1  vec_A =
2      -5.1423    6.1284         0
3  mag_A =
4      8.0000
5  mag_Y =
6      1
7  A_dot_Y =
8      -6.1284
9  theta_AY =
10     130
11 cross_ABnew =
12     18.3851    15.4269    94.6201
13 Y_dot_crossABnew =
14     -15.4269
15 mag_ABnew =
16     97.6164
17 theta_ABY =
18     99.0929

```

- Vector \vec{a} has a magnitude of 5.0 m and is directed east. Vector \vec{b} has a magnitude of 4.0 m and is directed 35° west of due north. What are (a) the magnitude and (b) the direction of $\vec{b} - \vec{a}$?

```

1  %% Reference: Question 3–46 from Fundamentals of Physics 10th Extended c2014
   ed. by Halliday, Resnick and Walker
2
3  clear all
4  close all
5  clc
6
7  theta = 90 - 35;
8  a = [5 0];
9  b = [-4*cosd(theta) 4*sind(theta)]
10 d = b - a
11 mag_d = sqrt(sum(d.*d))
12 theta_N = atand(d(2)./d(1))

```

```
13 theta_NW = 180 + theta_N
```

The outputs are as follows.

```
1 b =  
2     -2.2943     3.2766  
3 d =  
4     -7.2943     3.2766  
5 mag_d =  
6     7.9964  
7 theta_N =  
8     -24.1897  
9 theta_NW =  
10    155.8103
```

- Two vectors \vec{a} and \vec{b} have the components, in meters, $a_x = 3.2$, $a_y = 1.6$, $b_x = 0.50$, $b_y = 4.5$.
(a) Find the angle between the directions of \vec{a} and \vec{b} . There are two vectors in the xy plane that are perpendicular to \vec{a} and have a magnitude of 5.0 m. One, vector \vec{c} , has a positive x component and the other, vector \vec{d} , a negative x component. What are (b) the x component and (c) the y component of vector \vec{d} ?

```
1  %% Reference: Question 3–48 from Fundamentals of Physics 10th Extended c2014  
   ed. by Halliday, Resnick and Walker  
2  
3  clear all  
4  close all  
5  clc  
6  
7  theta = 90 - 35;  
8  a = [3.2 1.6];  
9  b = [0.5 4.5];  
10 mag_a = sqrt(sum(a.*a))  
11 mag_b = sqrt(sum(b.*b))  
12 a_dot_b = sum(a.*b)  
13 theta = acosd(a_dot_b / (mag_a * mag_b))  
14 d = 5;  
15 theta_a = atand(a(2)./a(1))  
16 theta_d = 90 + theta_a  
17 d_x = d*cosd(theta_d)  
18 d_y = d*sind(theta_d)  
19 vec_d = [d_x d_y]
```

The outputs are as follows.

```

1 mag_a =
2     3.5777
3 mag_b =
4     4.5277
5 a_dot_b =
6     8.8000
7 theta =
8     57.0948
9 theta_a =
10    26.5651
11 theta_d =
12    116.5651
13 d_x =
14    -2.2361
15 d_y =
16     4.4721
17 vec_d =
18    -2.2361     4.4721

```

3.2 Questions from One-Dimensional Motion

- A projectile's launch speed is five times its speed at maximum height. Find launch angle θ_0 .

```

1  %% Reference: Question 4–29 from Fundamentals of Physics 10th Extended c2014
   ed. by Halliday, Resnick and Walker
2
3  clear all
4  close all
5  clc
6
7  % To compute it numerically, I am assuming maximum velocity equals 1.
8  % However, you are free to choose any value. It will not affect the answer.
9  v_max = 1;
10 v_0 = 5*v_max;
11 theta = acosd(v_max/v_0)

```

And theta comes out to be 78.4630 in degrees.

- A soccer ball is kicked from the ground with an initial speed of 19.5 m s^{-1} at an upward angle of 45° . A player 55 m away in the direction of the kick starts running to meet the ball at that instant. What must be his average speed if he is to meet the ball just before it hits the ground?

```

1 %% Reference: Question 4–30 from Fundamentals of Physics 10th Extended c2014
  ed. by Halliday, Resnick and Walker
2
3 clear all
4 close all
5 clc
6
7 v_0 = 19.5;
8 theta_0 = 45.0;
9 x_player = 55.0;
10 g = 9.80;
11 t = (2*v_0*sind(theta_0))/g
12 x_ball = v_0*cosd(theta_0)*t
13 delta_x = x_ball - x_player
14 v_avg = delta_x/t

```

The outputs are as follows.

```

1 t =
2     2.8140
3 x_ball =
4     38.8010
5 delta_x =
6    -16.1990
7 v_avg =
8    -5.7566

```

- A lowly high diver pushes off horizontally with a speed of 2.00 m s^{-1} from the platform edge 10.0 m above the surface of the water. (a) At what horizontal distance from the edge is the diver 0.800 s after pushing off? (b) At what vertical distance above the surface of the water is the diver just then? (c) At what horizontal distance from the edge does the diver strike the water?

```

1 %% Reference: Question 4–37 from Fundamentals of Physics 10th Extended c2014
  ed. by Halliday, Resnick and Walker
2
3 clear all
4 close all
5 clc
6
7 v_0 = 2.0;
8 x_0 = 0.0;
9 y_0 = 10.0;

```

```
10 t = 0.8;
11 g = 9.8;
12 x = x_0 + (v_0 * t)
13 y = y_0 - (1/2)*g*t*t
14 t_new = sqrt((2*y_0)/g)
15 R = v_0 * t_new
```

The outputs are as follows.

```
1 x =
2   1.6000
3 y =
4   6.8640
5 t_new =
6   1.4286
7 R =
8   2.8571
```