# 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

# 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.

```
>> pi
ans =
    3.1416
>> format long
>> pi
ans =
   3.141592653589793
>> format short
>> pi
ans =
    3.1416
```

### 1.1.1 Vectors and Matrices

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

[1] Most 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*

```
newv =
     1     5    11     4     2    -1     6    -7    10    -7
>> u4 = u(4)                % Extracting the elements of a vector, i.e., u(4)
u4 =
     4
>> u=[1:6]                  % Generate a vector of equally-spaced elements with colon operator
u =
     1     2     3     4     5     6
>> u=[1:2:14]               % Increment by 2
u =
     1     3     5     7     9    11    13
>> transu = u'              % Get the tranpose of u
transu =
     1
     3
     5
     7
     9
    11
    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.

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

```
>> a = sin(45)          % Compute sine of 45 in radians
a =
    0.8509
>> b = sind(45)         % Compute sine of 45 in degrees
b =
    0.7071
>> c = cos(45)          % Compute cosine of 45 in radians
c =
    0.5253
>> d = cosd(45)         % Compute cosine of 45 in degrees
d =
    0.7071
>> e = tan(45)          % Compute tangent of 45 in radians
e =
    1.6198
>> f = tand(45)         % Compute tangent of 45 in degrees
f =
     1
>> g = acsc(45)         % Compute the cosecant-inverse of 45 in radians
g =
    0.0222
>> h = asec(45)         % Compute the secant-inverse of 45 in radians
h =
    1.5486
>> i = acotd(45)        % Compute the cotangent-inverse of 45 in degrees
i =
    1.2730
>> j = log(45)          % Compute the natural logarithm of 45
j =
    3.8067
>> k = log10(45)        % Compute the common logarithm of 45
k =
    1.6532
>> l = log2(45)         % Compute the logarithm in base 2 of 45
l =
    5.4919
>> m = exp(45)          % Compute the exponential of 45
m =
   3.4934e+19
>> n = sqrt(45)         % Compute the square root of 45
n =
    6.7082
```

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

### 1.1.3 Plotting in MATLAB

The command plot produces $2D$ graphics[2]. 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.

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

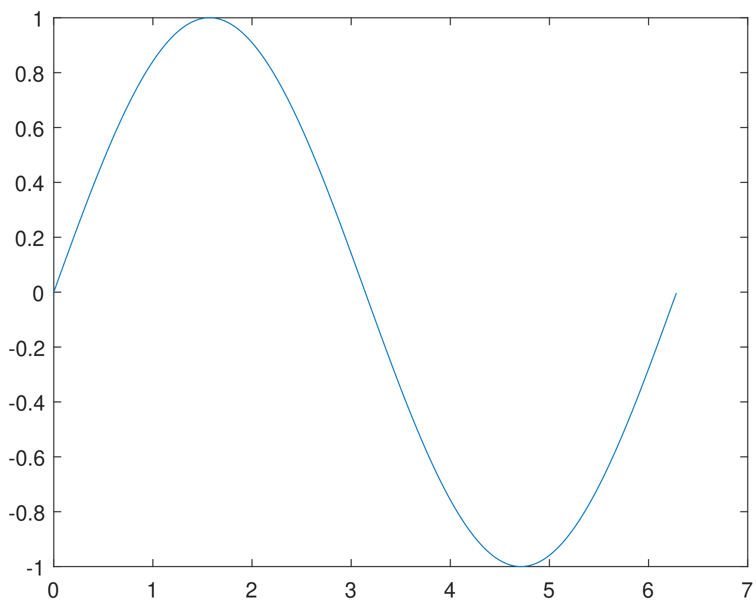


Figure 1.1: A sample 2D graph

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

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

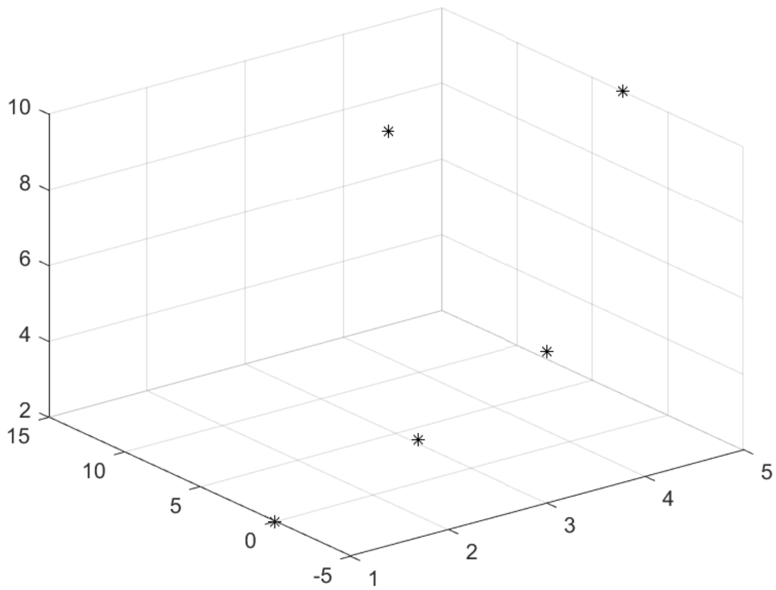


Figure 1.2: A sample 3D graph

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

```
x = -2:0.25:2;
y = x;
[X,Y] = meshgrid(x);
F = X.*exp(-X.^2-Y.^2);
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.

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

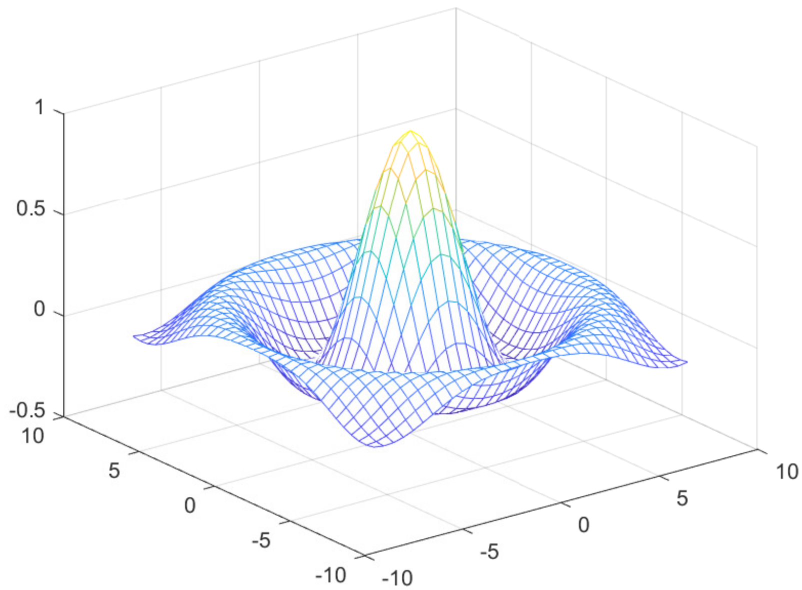


Figure 1.3: A sample mesh plot

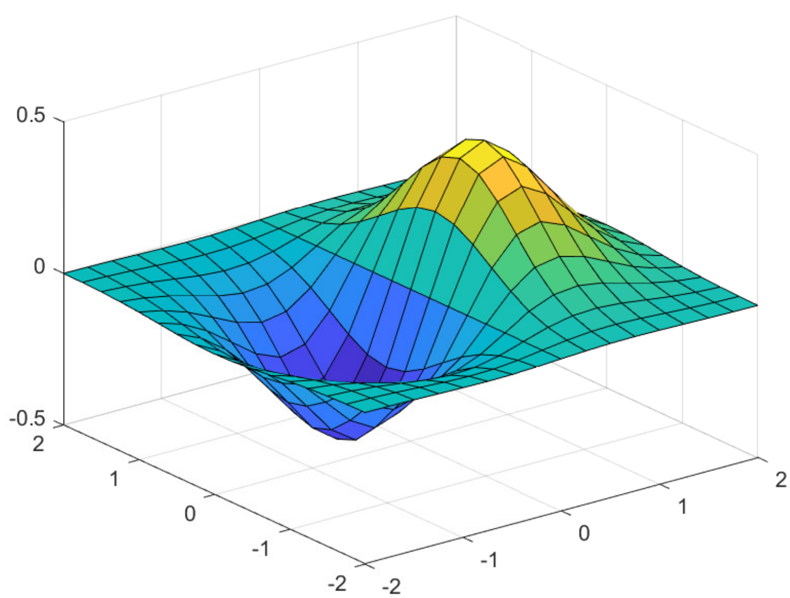


Figure 1.4: A sample surf plot

```
y2 = cos(x);
figure
plot(x,y1,x,y2)
title('Line Plot of Sine and Cosine Between −2\pi and 2\pi')
```

```
xlabel('−2\pi < x < 2\pi')
ylabel('Sine and Cosine Values')
legend({'y = sin(x)','y = cos(x)'})
```

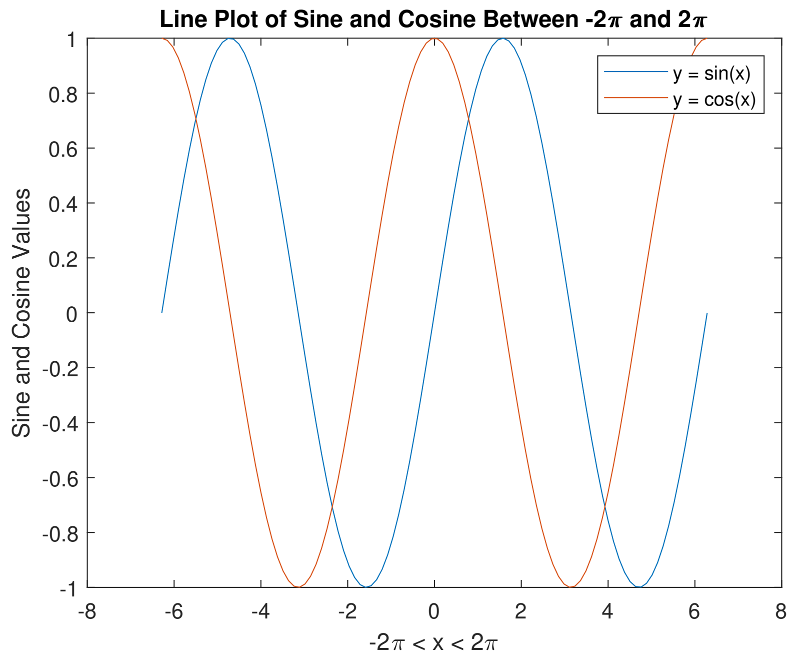


Figure 1.5: A sample plot with title and axes labeling

You can use the `subplot` command to obtain several smaller “subplots””in the same gure. 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 fth 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.

```
x = 0:0.01:5;
y = exp(−1.2*x).*sin(10*x+5);
subplot(1,2,1)
plot(x,y)
xlabel('x')
ylabel('y')
xlim([0 5])
ylim([−1 1])

```

```
x = -6:0.01:6;
y = abs(x.^3-100);
subplot(1,2,2)
plot(x,y)
xlabel('x')
ylabel('y')
axis([-6 6 0 350])
```

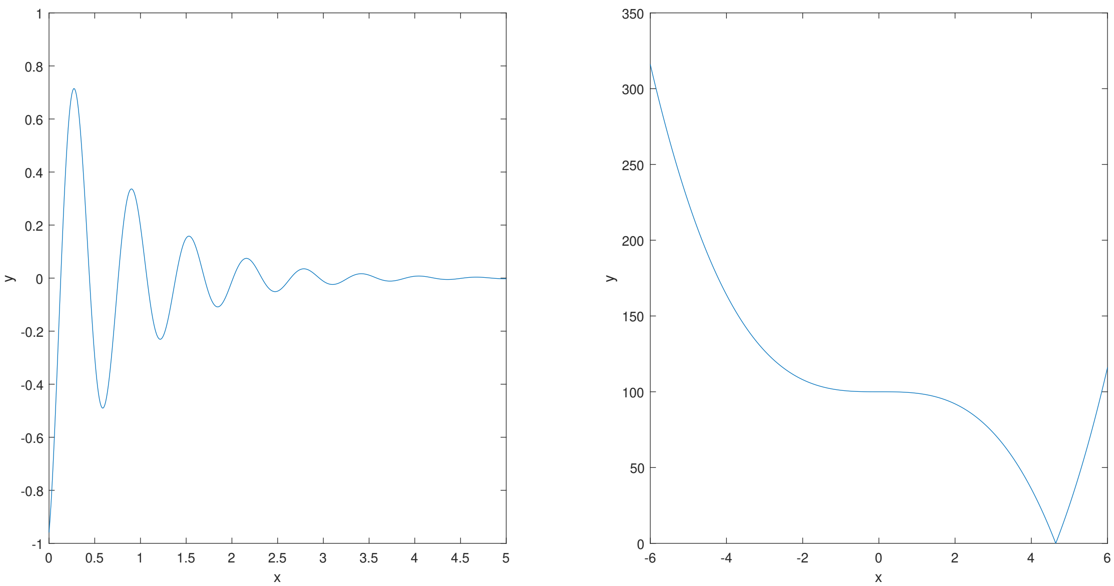


Figure 1.6: A sample subplot

### 1.1.4 Element–Wise Operations

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

```
% Let us start with an array x
x = [0, 0.25, 0.50, 0.75, 1.00];

% We would like to square each element of the array. In matlab, this can be
   done with:
% x.^2  % Notice the dot (.) in front of exponentiation (^).

x2 = x.^2
x2 =
         0    0.0625    0.2500    0.5625    1.0000
```

[3]Most 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 variable[3]. 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

```
%% Representation of Vectors

% List of numbers
% Enclose in square brackets
% Matlab treat every number as a vector

a = [1 2 3 4 5]
b = [6 7 8 9 10]

%% Semicolon ';' The semicolon cnn be used to construct arrays,
%  supress output from a MATLAB command,
%  or to separate commands entered on the same line.

% Addition of vectors
add = a + b

% Subtraction of vectors
sub = b - a

%% Multiplication of a vector by a scalar

g = 5*a
h = 4*b

% You can also check the following commands as well.
%(a) a + b
%(b) a* b
%(c) a*c
%(d) a.*d
```

*Palm III*

```
%(e) a.*b

%% norm can be used to find the magnitude of a vector

aNorm = norm(a)
bNorm = norm(b)

%% Dot Product of Real Vectors

A1 = [4 -1 2];
B1 = [2 -2 -1];

DotProduct = dot(A1,B1)

%% Cross  Product of Real Vectors

a1 = [1 2 3];
b1 = [4 5 6];

crossproduct = cross(a1,b1)

%% Angle theta

u = [1 2 0];
v = [1 0 0];

CosTheta = dot(u,v)/(norm(u)*norm(v))

%% theta in Degrees

ThetaInDegree = acosd(CosTheta)
```

And their outputs are shown below.

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

```
aNorm =
    7.4162
bNorm =
   18.1659
DotProduct =
     8
crossproduct =
    -3     6    -3
CosTheta =
    0.4472
ThetaInDegree =
   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.

```
>> a = [1 2 3; 4 5 6; 7 8 10]
a =
     1     2     3
     4     5     6
     7     8    10
>> aTrans = a'
aTrans =
     1     4     7
     2     5     8
     3     6    10
>> a3 = a.^3
a3 =
           1            8           27
          64          125          216
         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.

```
>> A = [a,a]
A =
     1     2     3     1     2     3
     4     5     6     4     5     6
     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.

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

## 1.3 Loops

To use MATLAB to solve many physics problems you have to know how to write loops[4]. 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 $\sum_{n=1}^{N} \frac{1}{n^2}$.

```
s = 0;      % set a variable to zero so that 1/n^2 can be repeatedly added
   to it
N = 10000;  % set the upper limit of the sum

for n=1:N   % start of the loop
    s = s + 1/n^2; % add 1/n^2 to s each time, then put the answer back into
        s
end         % end of the loop

fprintf('Sum = %g \n',s) % print the answer
```

And the `Sum` comes out to be `1.64483`.

[4]Most 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:

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

# Chapter 2

# Physics with MATLAB

## 2.1 Mechanics

### 2.1.1 Gravitational Force between Two Masses

```
% This program calculates and displays the Gravitational Force between two masses

% Input Variables
mass_1 = input('Enter the value of mass 1 (in kilogram) ');
mass_2 = input('Enter the value of mass 2 (in kilogram)');
distance = input('Enter the value of distance ');
G = 6.67*10^(-11); % Gravitational constant in the units of Nm^2/kg^2:

% Calculation
force = (G .* mass_1 .* mass_2)./(distance.^2); % computes the value  of Gravitational Force
display(['Force between the masses is = ',num2str(force),' newtons.'])
```

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

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

### 2.1.2 Gravitational Force–An Inverse Square Law

```
% We will show that gravitational force follows inverse square law.

% Input Variables
```

```
mass_1 = input('Enter the value of mass 1 (in kilogram) = ');
mass_2 = input('Enter the value of mass 2 (in kilogram) =  ');
r = input('Enter a positive value of distance (in meters) =  ');
G = 6.67*10^(-11);    % Gravitational constant in the units of Nm^2/kg^2

% Calculations
dr = r/200;           % calculates the Step Size
distance = r:dr:2*r;  % creates an array of 200 values
force = (G .* mass_1 .* mass_2)./(distance.^2);
plot(distance.^2,force,'LineWidth',1.5)
xlabel('Distance in meters^2')
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.

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

### 2.1.3 Free Fall Motion

```
% Input Variable:
% tfinal = final time (in seconds)
%
% Output Variables:
% t = array of times at which speed is % computed (in seconds)
% v = array of speeds (meters/second)

g = 9.81;             % Acceleration in SI units
tfinal = input('Enter final time (in seconds): ');
dt = tfinal/500;
t = 0:dt:tfinal;    % Creates an array of 501 time values
v = g*t;
plot(t,v)
xlabel('t in sec')
ylabel('v in m/s')
```

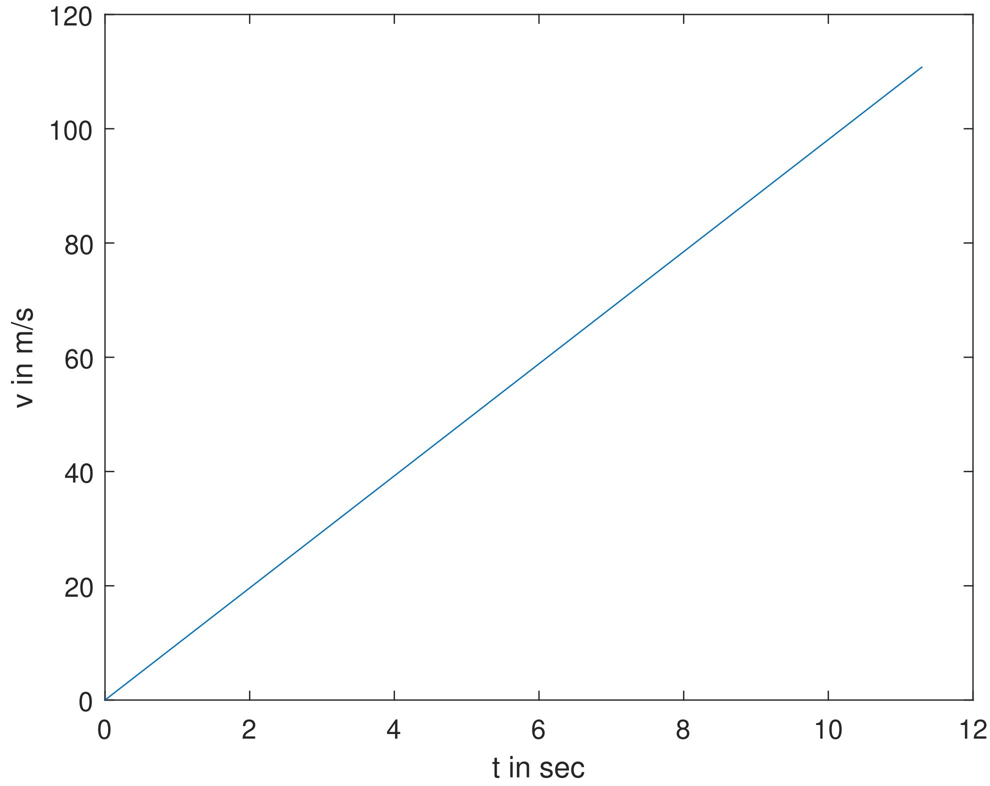


Figure 2.1: Falling objects in free fall (for t = 11.3 seconds)

### 2.1.4 Projectile Motion without making Custom Functions

```
%% Projectile's trajectory

x = 1:0.1:80;
g = 9.8;
v0 = 30;
theta = 30;
y = x .* tand(theta) - (x.^2 * g)/(2 * v0^2 * (cosd(theta)^2));
plot(x,y,'r','linewidth',1.5)

hold on
theta1 = 45;
y1 = x .* tand(theta1) - (x.^2 * g)/(2 * v0^2 * (cosd(theta1)^2));
plot(x,y1,'b','linewidth',1.5)

theta2 = 60;
y2 = x .* tand(theta2) - (x.^2 * g)/(2 * v0^2 * (cosd(theta2)^2));
plot(x,y2,'g','linewidth',1.5)

```

```
xlabel('x')
ylabel('y')
legend('\theta = 30^{o}','\theta = 45^{o}','\theta = 60^{o}')
hold off

%% Range of projectile

theta_new = 0:0.01:90;
R = (v0^2 * sind(2*theta_new))/g;
plot(theta_new,R,'r','linewidth',1.5)
xlabel('Angle (in degrees)')
ylabel('Range (in meters)')

%% Height of projectile

theta_new = 0:0.1:90;
H = (v0^2 * sind(theta_new).^2)/(2*g);
plot(theta_new,H,'r','linewidth',1.5)
xlabel('Angle (in degrees)')
ylabel('Height (in meters)')
```

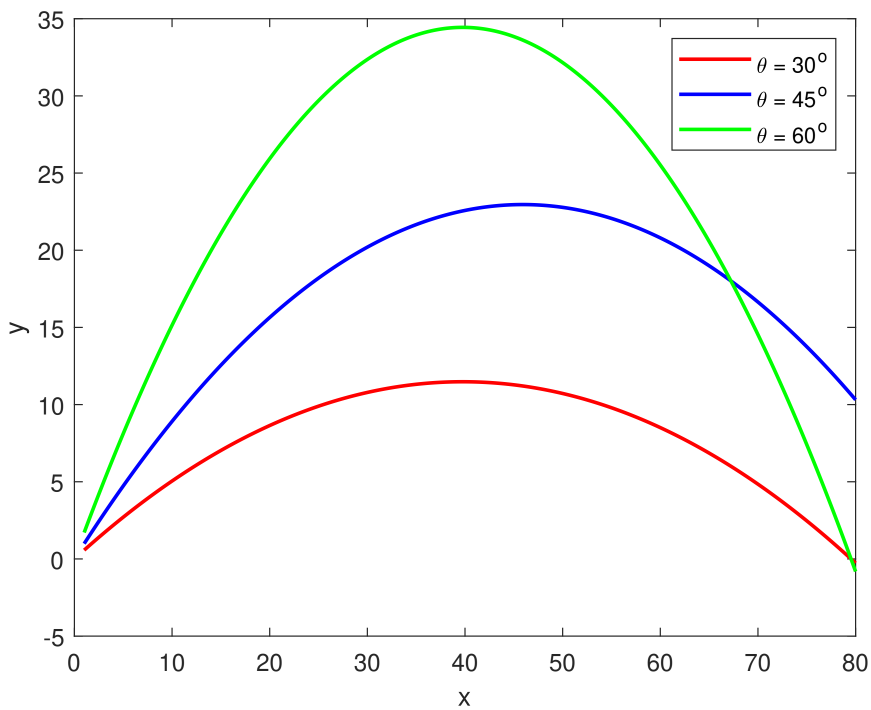


Figure 2.2: Trajectory of a projectile

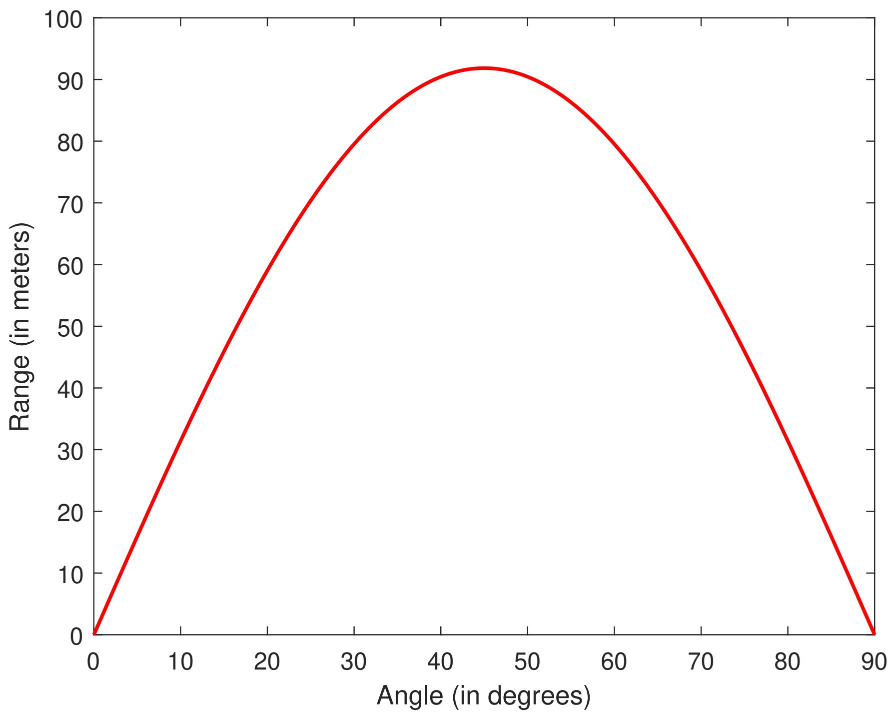


Figure 2.3: Range of a projectile

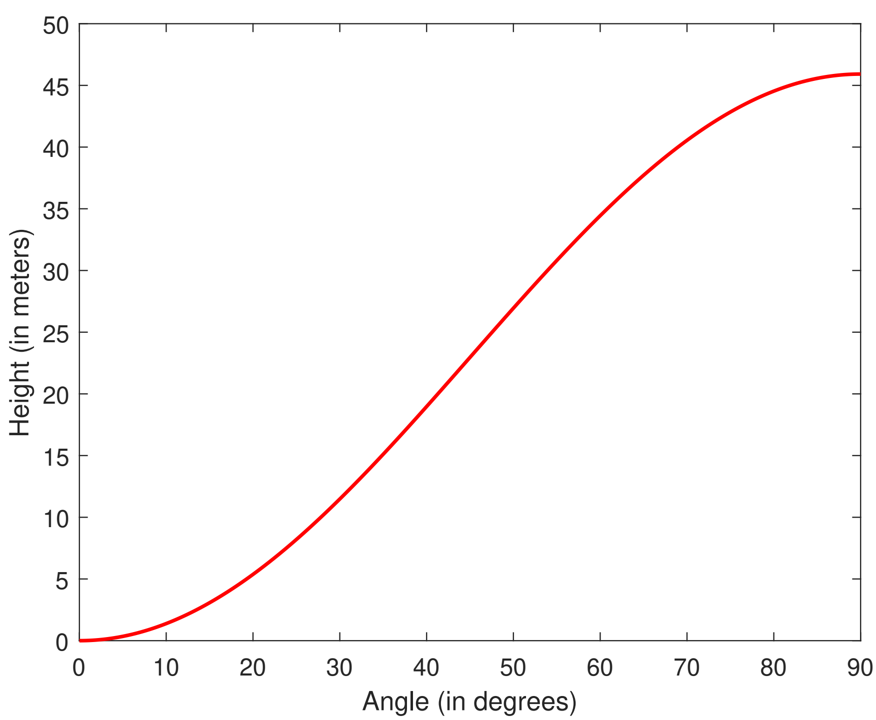


Figure 2.4: Height of a projectile

### 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.

```
%% Syntax
% Height = HProjectile(Angle of Incidence in degrees,velocity in m/s)
% File   = HProjectile.m

%% Function

function H = HProjectile(Theta,v)
H = (v.^2).*((sind(Theta)).^2)./(2.*(9.8));
end
```

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

```
%% Syntax
% Range = RProjectile(Angle of Incidence in degrees,velocity in m/s)
% File   = RProjectile.m

%% Function

function R = RProjectile(Theta,v)
R = (v.^2).*(sind(2.*Theta))./(9.8);
end
```

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

```
%% Syntax
% trProjectile(Angle of Incidence in degrees,velocity in m/s)
% File   = trProjectile.m

%% Function
function y = trProjectile(Theta,v)
x = 0:0.1:RProjectile(Theta,v);

for i = 1:length(x)
    y(i) = tand(Theta).*x(i) - ((9.8).*x(i).^2)./(2.*(v.*cosd(Theta)).^2);
end
```

```
plot(x,y,'linewidth',1.5);
grid on
xlabel('Range (m)')
ylabel('Altitude (m)')
```

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

```
>> Theta = 45;
>> v = 10;
>> H = HProjectile(Theta,v)
H =
    2.5510
>> R = RProjectile(Theta,v)
R =
   10.2041
>> Y = trProjectile(Theta,v);
```

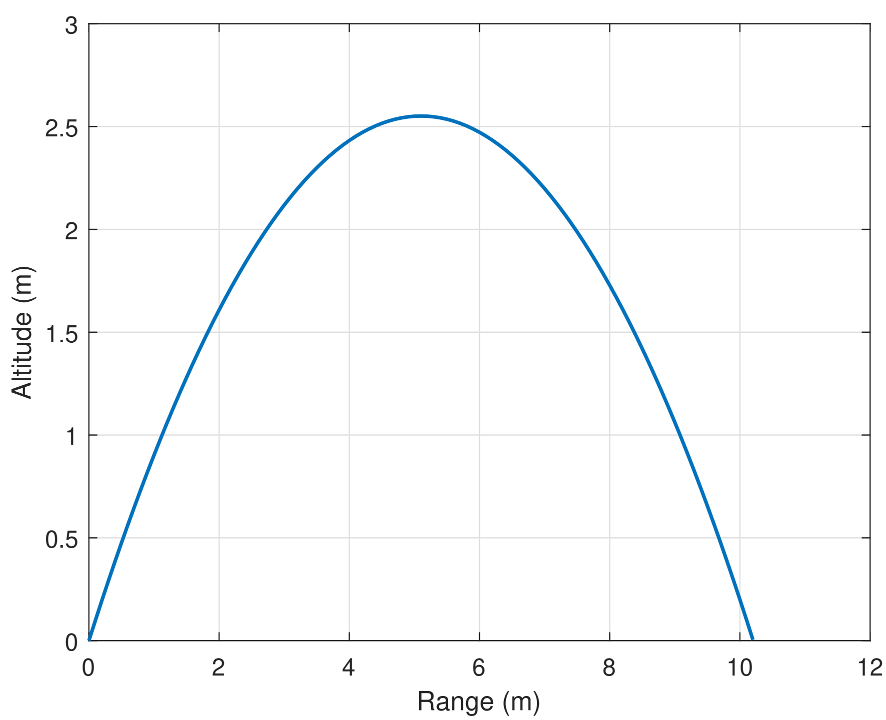


Figure 2.5: Trajectory of a projectile (using custom functions)

## 2.2 Waves and Oscillations

### 2.2.1 SHM as Circular Motion

```
%% This code will demonstrate from equations of motion of SHM that SHM is a type of circular motion.
%  Code starts from here.

clear all       % Clears the workspace
close all       % Close all previous plots and figures
clc             % Clears the command window


N = 1080;       % Declare total phase of oscillation
A0 = 10;        % Declare amplitude
theta(1) = 0;   % Initial value of theta


for i=1:N;      % For loop to find phase angle
% theta(i) is the previous angle and theta(i+1) in the new angle.
% theta is measured in radians.
    theta(i+1) = theta(i) + 1;
% Convert theta(i+1) into degrees by multiplying it with (pi/180).
% alpha(i) is the new theta(i+1) measured in degrees.
    alpha(i) = (pi/180) * theta(i+1);
end


% In the next three for loops, x represents first simple harmonic
% oscillator and y represents second simple harmonic oscillator


% For loop to find position of both SHOs
for i=1:N;
    x(i) = A0 * cos(alpha(i));     % Position equation of first SHO
    y(i) = A0 * sin(alpha(i));     % Position equation of second SHO
end


% For loop to find velocity of both SHOs
for i=1:N;
    vx(i) = - A0 * sin(alpha(i));  % Velocity equation of first SHO
    vy(i) = A0 * cos(alpha(i));    % Velocity equation of second SHO
end
```

```
% For loop to find acceleration of both SHOs
for i=1:N;
    ax(i) = - A0 * cos(alpha(i));  % Acceleration equation of first SHO
    ay(i) = - A0 * sin(alpha(i));  % Acceleration equation of second SHO
end


% Let's plot Phase Angle vs. Position
figure(1);
hold on     % hold ON sets the NextPlot property of the current figure and axes to add.
    plot(alpha,x,'linewidth',1.5)   % LineWidth sets the width of line.
    plot(alpha,y,'linewidth',1.5)
    xlabel('Phase Angle')   % xlabel('text') adds text beside the X-axis.
    ylabel('Amplitude')     % ylabel('text') adds text beside the Y-axis.
    ylim([-12 12])          % ylim([YMIN YMAX] sets the y limits.
    legend('x(i)','y(i)')
    title('Phase Angle vs. Position (in SHM)')
    grid on                 % grid ON adds major grid lines.
hold off    % hold OFF sets the NextPlot property of the current figure and axes to replace.


% Let's plot Phase Angle vs. Velocity
figure(2);
hold on
    plot(alpha,vx,'linewidth',1.5)
    plot(alpha,vy,'linewidth',1.5)
    xlabel('Phase Angle')
    ylabel('Velocity')
    ylim([-12 12])
    legend('v_{x}(i)','v_{y}(i)')
    title('Phase Angle vs. Velocity (in SHM)')
    grid on
hold off


% Let's plot Phase Angle vs. Acceleration
figure(3);
hold on
    plot(alpha,ax,'linewidth',1.5)
    plot(alpha,ay,'linewidth',1.5)
    xlabel('Phase Angle')
```

```
    ylabel('Acceleration')
    ylim([-12 12])
    legend('a_{x}(i)','a_{y}(i)')
    title('Phase Angle vs. Acceleration (in SHM)')
    grid on
hold off


% Let's plot to show that both SHOs depict circular motion
figure(4);
    plot(x,y,'linewidth',1.5)
    xlabel('x(t)')
    ylabel('y(t)')
    xlim([-12 12])
    ylim([-12 12])
    title('Simple Harmonic Motion as Circular Motion')
    grid on


% Let's combine first three plots into one.
figure;
subplot(3,1,1)          % Write help subplot in Command Window to understand it
title('Phase Angle vs. Position, Velocity and Acceleration (in SHM)');
hold on
    plot(alpha,x,'linewidth',1.5)
    plot(alpha,y,'linewidth',1.5)
    ylabel('Amplitude')
    ylim([-12 12])
    legend('x(i)','y(i)')
    grid on
hold off


subplot(3,1,2)
hold on
    plot(alpha,vx,'linewidth',1.5)
    plot(alpha,vy,'linewidth',1.5)
    ylabel('Velocity')
    ylim([-12 12])
    legend('v_{x}(i)','v_{y}(i)')
    grid on
hold off


```

```
subplot(3,1,3)
hold on
    plot(alpha,ax,'linewidth',1.5)
    plot(alpha,ay,'linewidth',1.5)
    xlabel('Phase Angle')
    ylabel('Acceleration')
    ylim([-12 12])
    legend('a_{x}(i)','a_{y}(i)')
    grid on
hold off

%% Task completed.
```

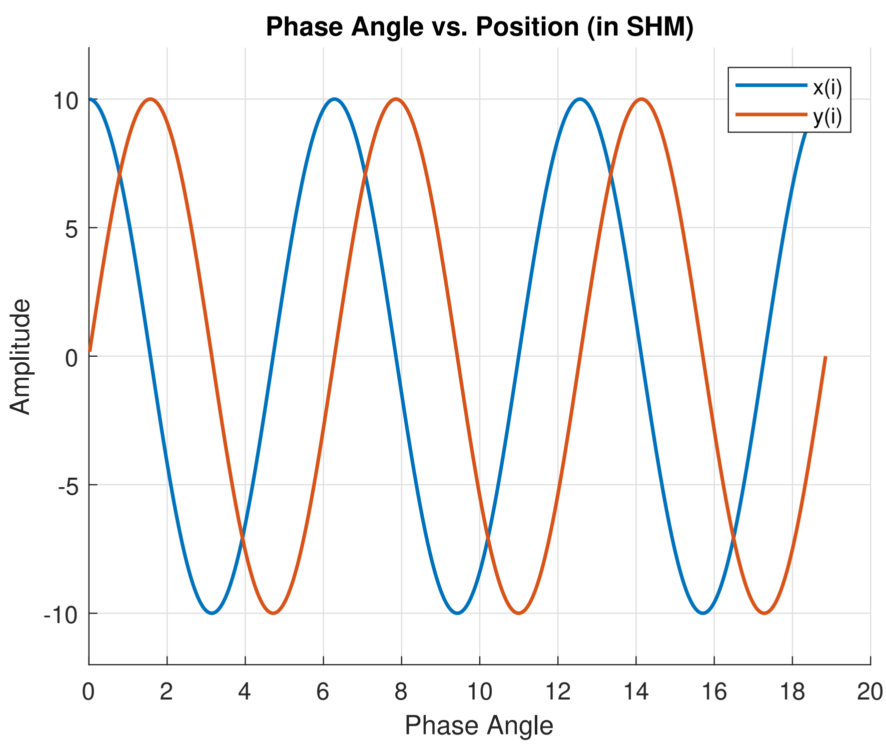


Figure 2.6: Phase Angle vs. Position

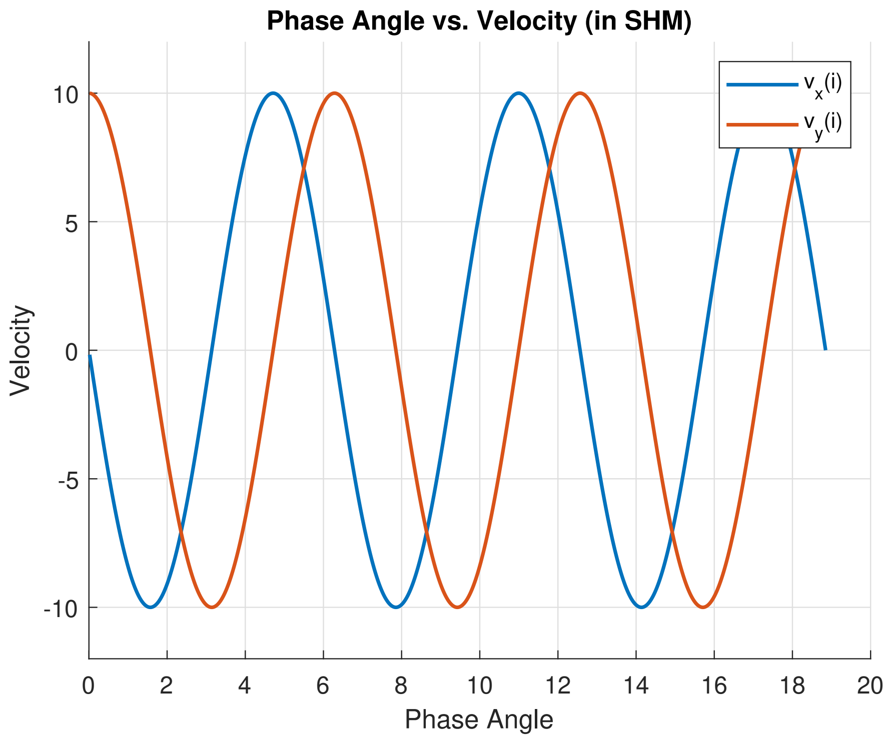


Figure 2.7: Phase Angle vs. Velocity

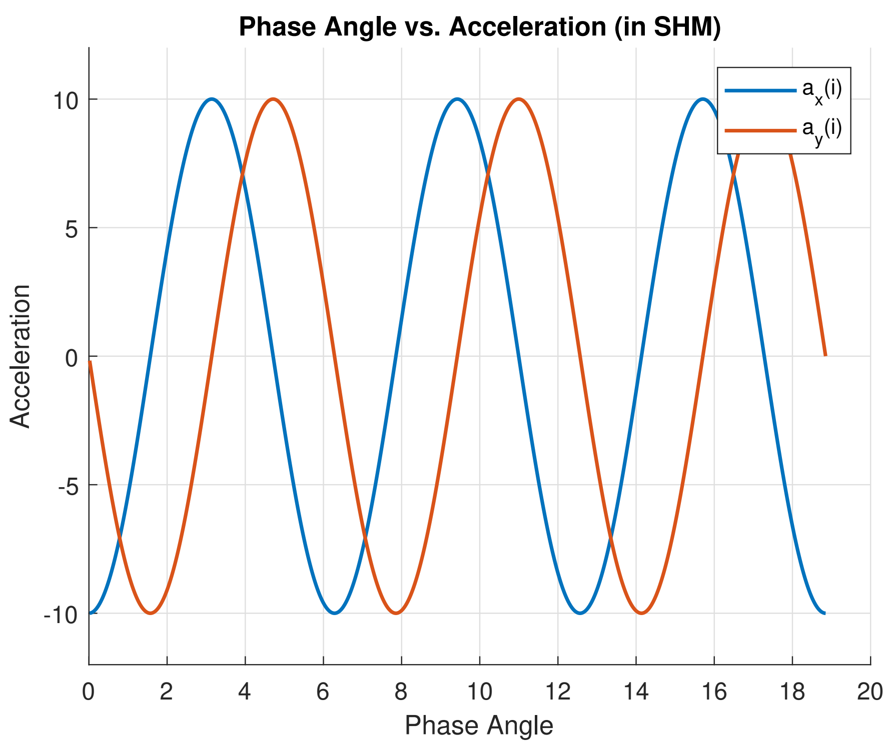


Figure 2.8: Phase Angle vs. Acceleration

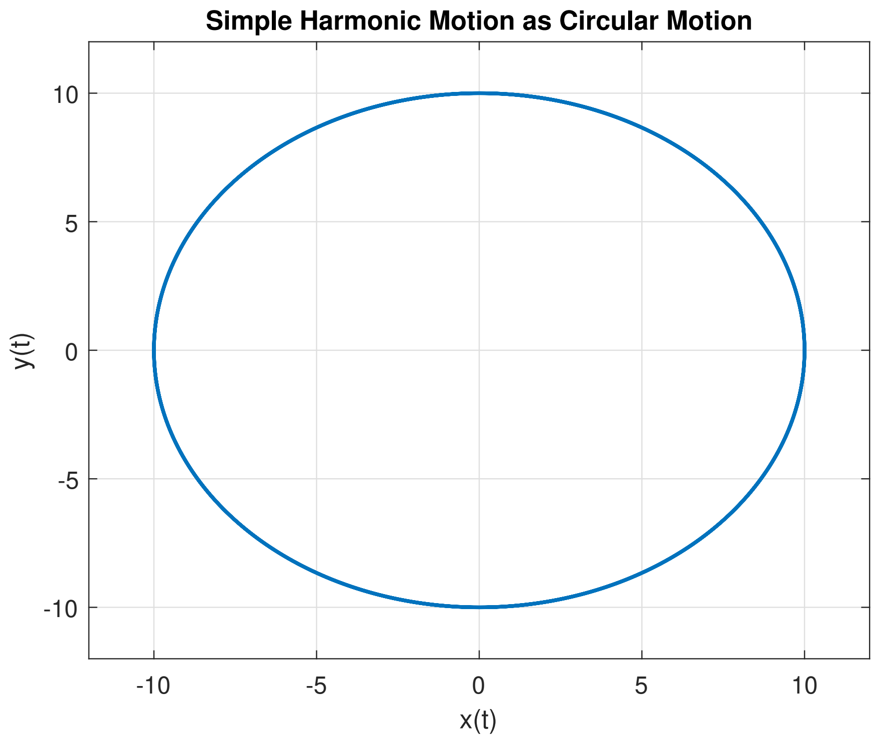


Figure 2.9: SHM as circular motion

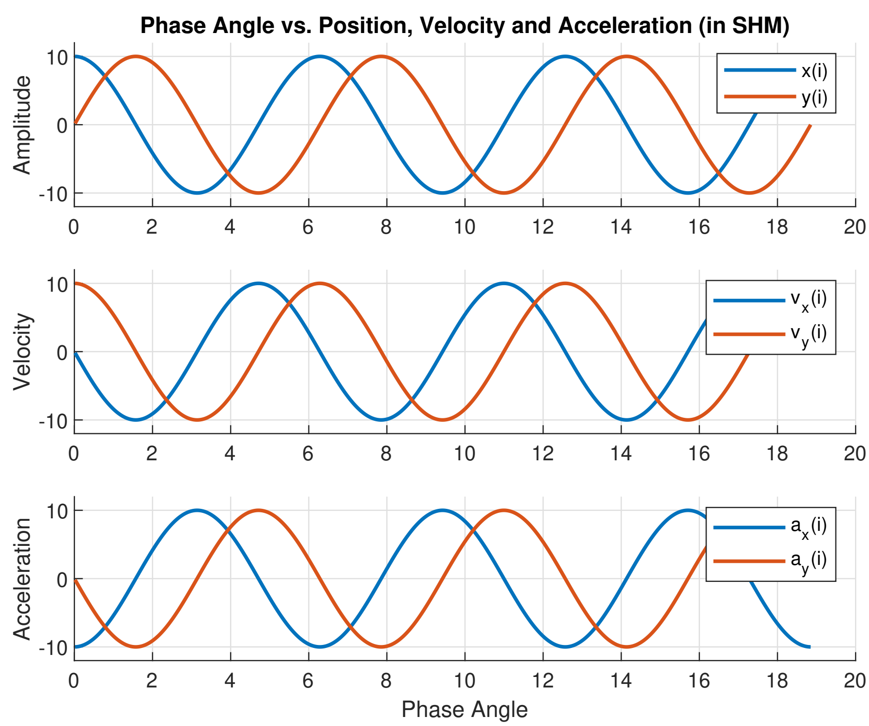


Figure 2.10: Phase Angle vs. Position, Velocity and Acceleration

## 2.3 Electricity and Magnetism

### 2.3.1 Coulomb Force between Charges

```
%% Coulomb law for two point charges

eps0 = 8.854e-12;
kC = 1/(4*pi*eps0);
q1 = -1e-13;
q2 = +1e-10;
r  = [-12:0.1:12].*1e-12;

F  = (kC*q1*q2)./(r.*r);

plot(r,F,'LineWidth',1.5)
xlabel('Distance (in meters)')
ylabel('Coulomb force (in newtons)')
```

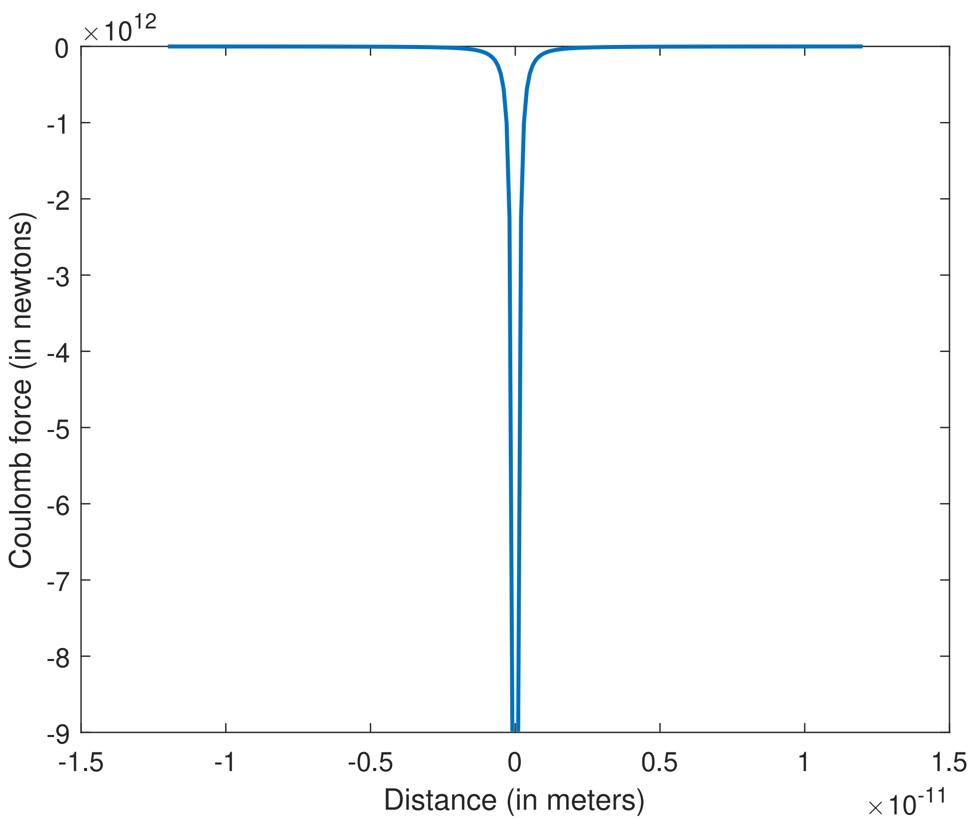


Figure 2.11: Electrostatic (Coulomb) force as inverse square law

### 2.3.2 Coulomb Force between Charges using For Loop

```
%% Cforce - Program to compute Coulomb force between charges
%  Source: https://ualr.edu/dcwold/phys2322/cforce/cforce.html

clear all;  help Cforce;    % Clear memory; print header

%@ Enter your username
fprintf('Enter your username (userid); \n');
fprintf('USERNAME: ABName  \n');
Username = input('  USERNAME: ','s'); % Read input as a text string
fprintf('\n');

%@ Initialize variables (e.g., positions of charges, physical constants)
NCharges = input('Enter the number of charges: ');
for iCharge=1:NCharges
  fprintf('------ \n  For charge #%g \n',iCharge);
  r_in = input('Enter position (in m) as [x y]: ');
  x(iCharge) = r_in(1);   % x-component of position
  y(iCharge) = r_in(2);   % y-component of position
  q(iCharge) = input('Enter charge (in C): ');
end

%@ Find xmin, xmax, ymin, and ymax
xmin = min(x)-1;
ymin = min(y)-1;
xmax = max(x)+1;
ymax = max(y)+1;

Epsilon0 = 8.85e-12;    % Permittivity of free space (C^2/(N m^2))
Constant = 1/(4*pi*Epsilon0);  % Useful constant

%@ Loop over charges to compute the force on each charge
fprintf('\n\n Forces are: \n\n');
for iCharge = 1:NCharges

  Fx = 0.0;  % Initialize components of total force to zero
  Fy = 0.0;

  %@ Loop over other charges to compute force on this charge
  for jCharge = 1:NCharges
 if( iCharge ~= jCharge ) % If iCharge NOT equal to jCharge

   %@ Compute the components of vector distance between two charges
   xij = x(iCharge) - x(jCharge);
```

```
   yij = y(iCharge) - y(jCharge);
   Rij = sqrt(xij^2 + yij^2);

   %@ Compute the x and y components of the force between
     %@ these two charges using Coulomb's law

   Fx = Fx + Constant*q(iCharge)*q(jCharge)*xij/Rij^3;
   Fy = Fy + Constant*q(iCharge)*q(jCharge)*yij/Rij^3;

    end
 end
  Fxnet(iCharge) = Fx;
  Fynet(iCharge) = Fy;
  %@ Print out the total force on this charge due to the others
  fprintf('Force on charge #%g is: \n',iCharge);
  fprintf(' x-component: %g N \n',Fx);
  fprintf(' y-component: %g N \n',Fy);
end

%@ Plot position of charges
clf; % Clear graphics figure window
figure;     % Bring figure window forward
plot(x,y,'bo');
axis([xmin xmax ymin ymax]);
for j = 1:NCharges
   text(x(j),y(j),sprintf('  %g',j));
end
%@ Add force direction to position of charges
hold on;
quiver(x,y,Fxnet,Fynet,'r');    % Draw arrows for force
title([Username,',   ',date,', ','Cforce: Position of charges and direction of forces']);
xlabel('x (m)'); ylabel('y (m)');
hold off;
```

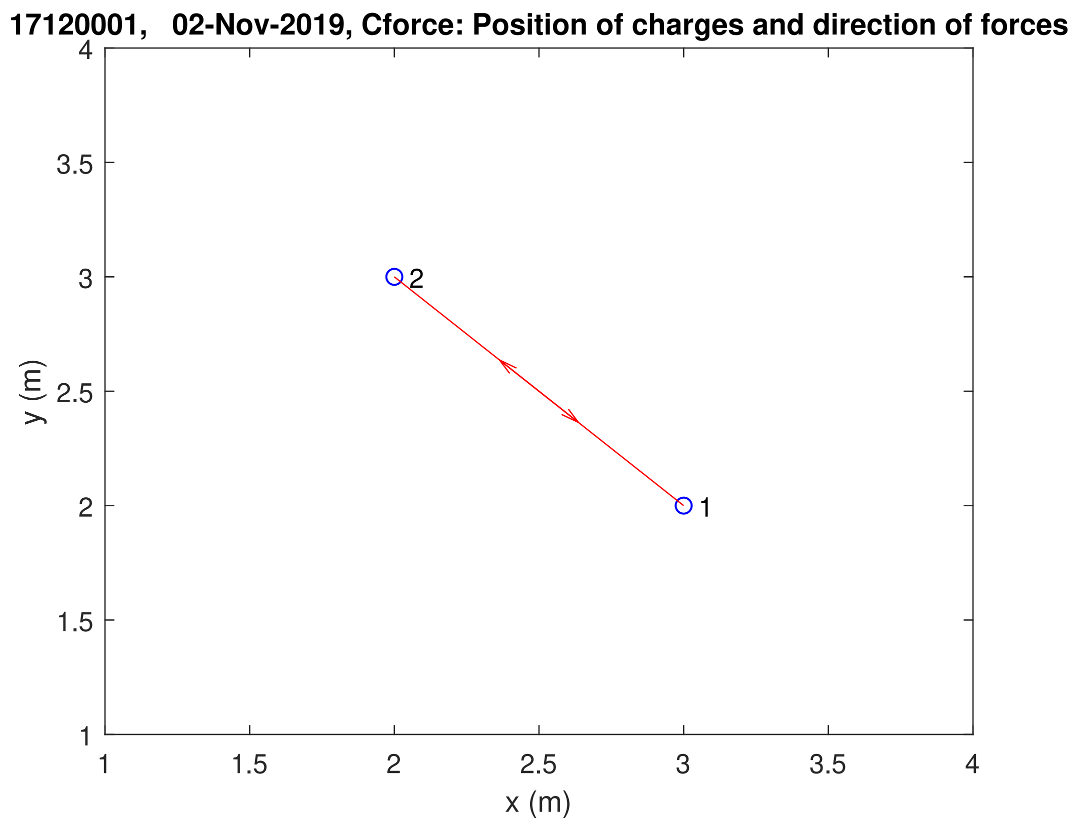


Figure 2.12: Electrostatic (Coulomb) force as inverse square law

### 2.3.3 Fields due to Discrete and Line Charge Distributions

```
%% Fields due to discrete and line charge distributions
%  Author: S. Mandayam, ECE, Rowan University

close all;
clear;

[x,y,z] = meshgrid(-1:0.05:1,0.001:0.05:1,-1:0.05:1);
% [X,Y,Z] = meshgrid(x,y,z) returns 3-D grid coordinates defined by the
% vectors x, y, and z. The grid represented by X, Y, and Z has size
% length(y)-by-length(x)-by-length(z).

% Point Charge
E = 1./(x.^2+y.^2+z.^2);

figure(1);
slice(x,y,z,log(E),[-0.9:0.05:0.9],0.9,[-0.9:0.05:0.9]);
% slice(X,Y,Z,V,xslice,yslice,zslice) draws slices for the volumetric data
% V. Specify X,Y, and Z as the coordinate data. Specify xslice, yslice, and
```

```
% zslice as the slice locations using one of these forms:
shading interp;
% shading interp varies the color in each line segment and face by
% interpolating the colormap index or true color value across the line or
% face.
colormap hsv;
% colormap map sets the colormap for the current figure to one of the
% predefined colormaps. If you set the colormap for the figure, then axes
% and charts in the figure use the same colormap
xlabel('x');
ylabel('y');
zlabel('z');
title('Electric Field (Log Magnitude) due to point charge at origin (0,0,0)');
axis square;
colorbar;
% colorbar displays a vertical colorbar to the right of the current axes or
% chart. Colorbars display the current colormap and indicate the mapping of
% data values into the colormap.
rotate3d on;
% rotate3d on turns on rotate mode and enables rotation on all axes within
% the current figure.
pause;
% pause temporarily stops MATLAB execution and waits for the user to press
% any key.

% Line Charge
E = 1./sqrt(x.^2+y.^2);
figure(2);
slice(x,y,z,log(E),[-0.9:0.1:0.9],0.9,[-0.9:0.1:0.9]);
shading interp;
colormap hsv;
xlabel('x');
ylabel('y');
zlabel('z');
title('Electric Field (Log Magnitude) due to line charge along z-axis');
axis square;
colorbar;
rotate3d on;
```

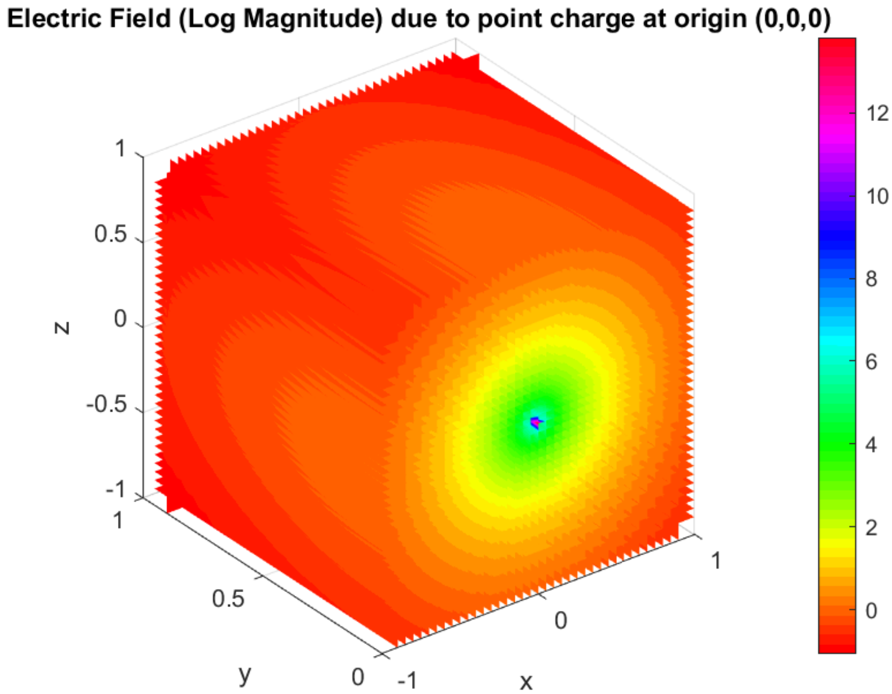


Figure 2.13: Electric Field (Log Magnitude) due to point charge at origin (0,0,0)

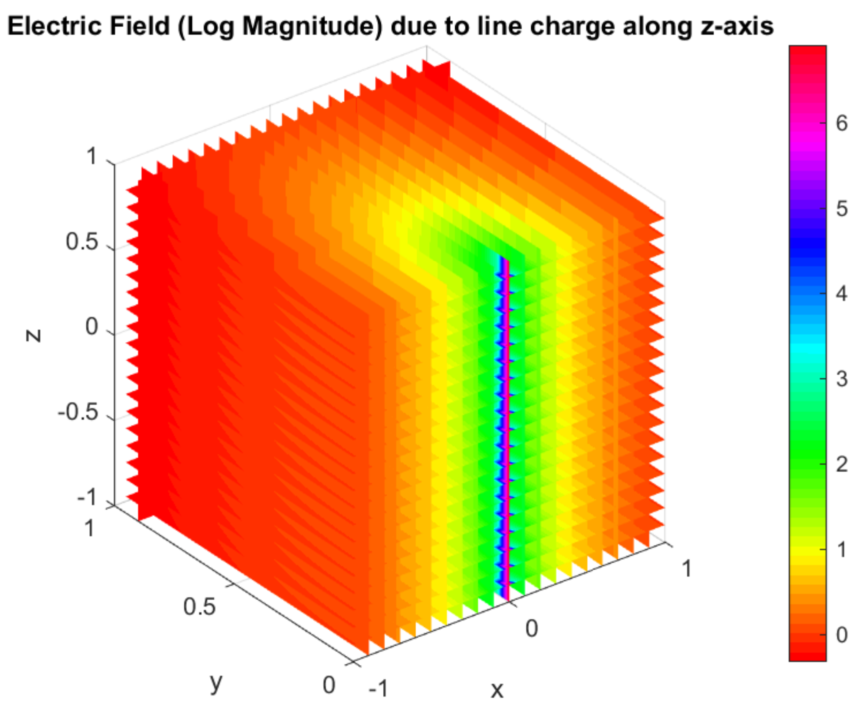


Figure 2.14: Electric Field (Log Magnitude) due to line charge along z–axis

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

```
%--------------------------------------------------------------------------%
%       This simple program computes the Electric Fields due to dipole
%           in a 2-D plane using the Coulomb's Law
%--------------------------------------------------------------------------%

%--------------------------------------------------------------------------%
%                  REFERENCE
% SADIKU, ELEMENTS OF ELECTROMAGNETICS, 4TH EDITION, OXFORD
%--------------------------------------------------------------------------%


clc
close all; clear all;

%--------------------------------------------------------------------------%
%                   SYMBOLS USED IN THIS CODE
%--------------------------------------------------------------------------%

% E = Total electric field
% Ex = X-Component of Electric-Field
% Ey = Y-Component of Electric-Field
% n = Number of charges
% Q = All the 'n' charges are stored here
% Nx = Number of grid points in X- direction
% Ny = Number of grid points in Y-Direction
% eps_r = Relative permittivity
% r = distance between a selected point and the location of charge
% ex = unit vector for x-component electric field
% ey = unit vector for y-component electric field
%--------------------------------------------------------------------------%


%--------------------------------------------------------------------------%
%                          INITIALIZATION
%          Here, all the grid, size, charges, etc. are defined
%--------------------------------------------------------------------------%

% Constant 1/(4*pi*epsilon_0) = 9*10^9
k = 9*10^9;

% Enter the Relative permittivity
eps_r = 1;
charge_order = 10^-9; % milli, micro, nano etc..
```

```
const = k*charge_order/eps_r;

% Enter the dimensions
Nx = 101; % For 1 meter
Ny = 101; % For 1 meter

% Enter the number of charges.
n = 2;

% Electric fields Initialization
E_f = zeros(Nx,Ny);
Ex = E_f;
Ey = E_f;

% Vectors initialization
ex = E_f;
ey = E_f;
r = E_f;
r_square = E_f;

% Array of charges
Q = [1,-1];

% Array of locations
X = [5,-5];
Y = [0,0];

%-------------------------------------------------------------------------%
%                       COMPUTATION OF ELECTRIC FIELDS
%-------------------------------------------------------------------------%

%  Repeat for all the 'n' charges
for k = 1:n
    q = Q(k);

    % Compute the unit vectors
    for i=1:Nx
        for j=1:Ny

            r_square(i,j) = (i-51-X(k))^2+(j-51-Y(k))^2;
            r(i,j) = sqrt(r_square(i,j));
            ex(i,j) = ex(i,j)+(i-51-X(k))./r(i,j);
            ey(i,j) = ey(i,j)+(j-51-Y(k))./r(i,j);
        end
    end
```

```
    E_f = E_f + q.*const./r_square;

    Ex = Ex + E_f.*ex.*const;
    Ey = Ex + E_f.*ey.*const;

end

%--------------------------------------------------------------------------%
%                          PLOT THE RESULTS
%--------------------------------------------------------------------------%

x_range = (1:Nx)-51;
y_range = (1:Ny)-51;
contour_range = -8:0.02:8;
contour(x_range,y_range,E_f',contour_range,'linewidth',0.7);
axis([-15 15 -15 15]);
colorbar('location','eastoutside','fontsize',12);
xlabel('x ','fontsize',14);
ylabel('y ','fontsize',14);
title('Electric field distribution, E (x,y) in V/m','fontsize',14);
```

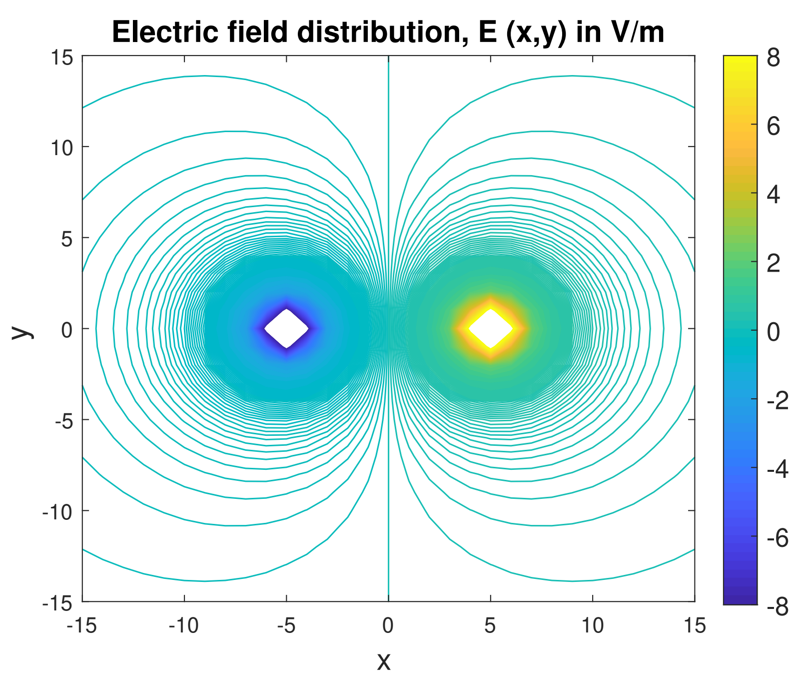


Figure 2.15: Electric field distribution of a dipole

### 2.3.5 Electric Field due to a Point Charge

```
k = 9e9;
r = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20];
E = zeros(20);
q = input('Enter Charge: ');

for i = 1:20
    E(i) = (k*q)/(r(i)^2);
end

plot(r,E)
xlabel('Distance (m)')
ylabel('Electric Field (N/C)')
```

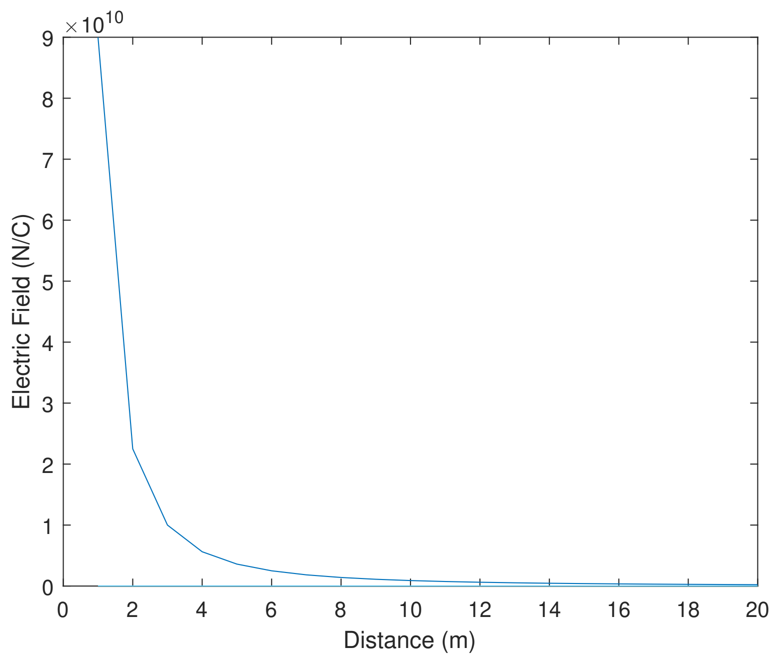


Figure 2.16: Electric Field due to a Point Charge

### 2.3.6 Electric Field due to a Dipole

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

```
for i = 1:20
    E(i) = (k*q*d)/(z(i)^3);
end

plot(z,E)
xlabel('Distance (m)')
ylabel('Electric Field (N/C)')
```

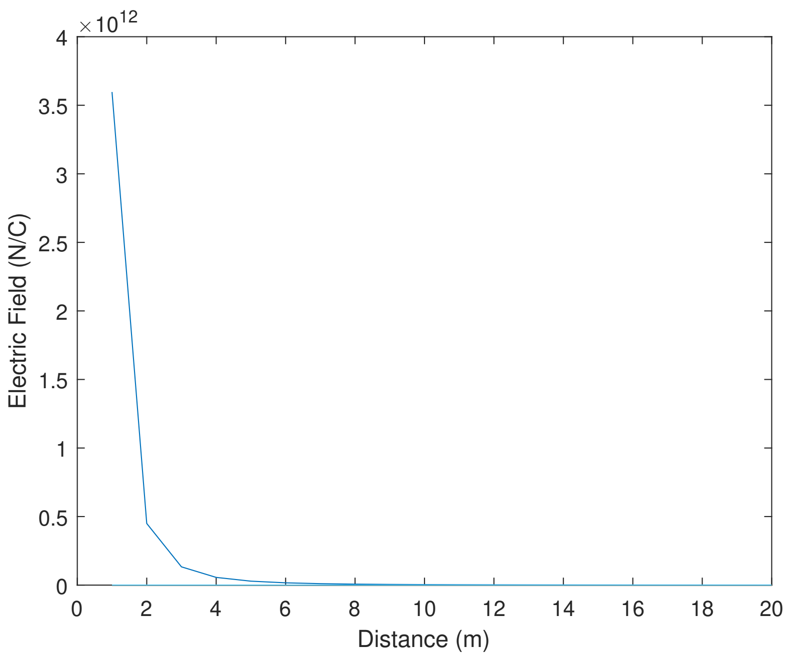


Figure 2.17: Electric Field due to a Dipole

### 2.3.7 Electric Potential due to a Point Charge

```
k = 9e9;
r = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20];
V = zeros(20);
q = input('Enter Charge: ');

for i = 1:20
    V(i) = (k*q)/(r(i));
end

plot(r,V)
xlabel('Distance (m)')
ylabel('Electric Potential (V)')
```

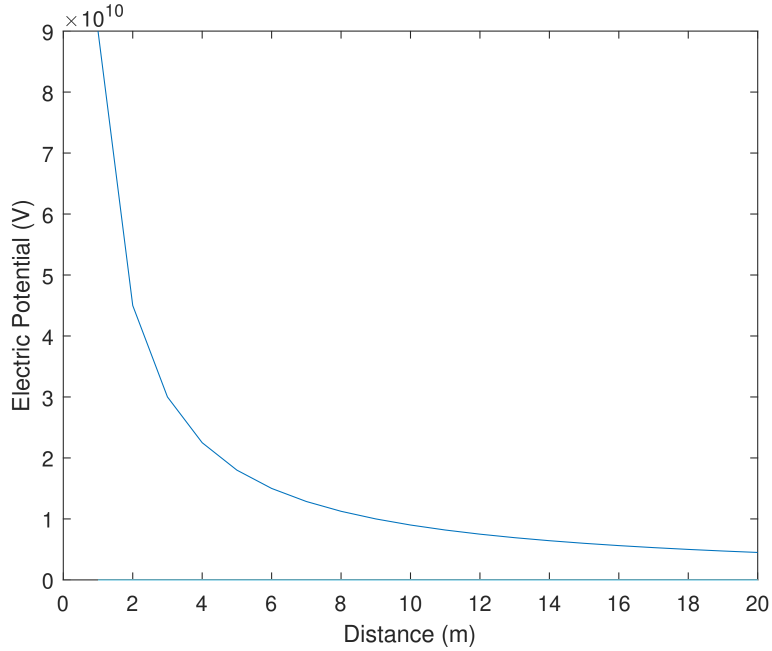


Figure 2.18: Electric Potential due to a Point Charge

# 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 $\vec{A} \times (\vec{B} + 3.00\hat{k})$?

```
%% Reference: Question 4-47 from Fundamentals of Physics 10th Extended c2014 ed. by Halliday, Resnick and Walker

clear all
close all
clc

A = 8.0;
theta_A = 130;

vec_A = [A*cosd(theta_A) A*sind(theta_A) 0]

vec_B = [-7.72 -9.20 0];
vec_Y = [0 -1 0];

mag_A = sqrt(sum(vec_A.*vec_A))
mag_Y = sqrt(sum(vec_Y.*vec_Y))

A_dot_Y = sum(vec_A.*vec_Y)
theta_AY = 270 - acosd(A_dot_Y / (mag_A * mag_Y))

```

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

The outputs are as follows.

```
vec_A =
   -5.1423    6.1284         0
mag_A =
    8.0000
mag_Y =
     1
A_dot_Y =
   -6.1284
theta_AY =
   130
cross_ABnew =
   18.3851   15.4269   94.6201
Y_dot_crossABnew =
  -15.4269
mag_ABnew =
   97.6164
theta_ABY =
   99.0929
```

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

```
%% Reference: Question 3-46 from Fundamentals of Physics 10th Extended c2014 ed. by Halliday, Resnick and Walker

clear all
close all
clc

theta = 90 - 35;
a = [5 0];
b = [-4*cosd(theta) 4*sind(theta)]
d = b - a
mag_d = sqrt(sum(d.*d))
theta_N = atand(d(2)./d(1))
```

```
theta_NW = 180 + theta_N
```

The outputs are as follows.

```
b =
   -2.2943    3.2766
d =
   -7.2943    3.2766
mag_d =
    7.9964
theta_N =
  -24.1897
theta_NW =
  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 $5.0\,\mathrm{m}$. 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 $\vec{d}$?

```
%% Reference: Question 3-48 from Fundamentals of Physics 10th Extended c2014
    ed. by Halliday, Resnick and Walker

clear all
close all
clc

theta = 90 - 35;
a = [3.2 1.6];
b = [0.5 4.5];
mag_a = sqrt(sum(a.*a))
mag_b = sqrt(sum(b.*b))
a_dot_b = sum(a.*b)
theta = acosd(a_dot_b / (mag_a * mag_b))
d = 5;
theta_a = atand(a(2)./a(1))
theta_d = 90 + theta_a
d_x = d*cosd(theta_d)
d_y = d*sind(theta_d)
vec_d = [d_x d_y]
```

The outputs are as follows.

```
mag_a =
    3.5777
mag_b =
    4.5277
a_dot_b =
    8.8000
theta =
   57.0948
theta_a =
   26.5651
theta_d =
  116.5651
d_x =
   -2.2361
d_y =
    4.4721
vec_d =
   -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 $\theta_0$.

```
%% Reference: Question 4-29 from Fundamentals of Physics 10th Extended c2014
    ed. by Halliday, Resnick and Walker

clear all
close all
clc

% To compute it numerically, I am assuming maximum velocity equals 1.
% However, you are free to choose any value. It will not affect the answer.
v_max = 1;
v_0 = 5*v_max;
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 $19.5\,\mathrm{m\,s^{-1}}$ at an upward angle of $45°$. A player $55\,\mathrm{m}$ 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?

```
%% Reference: Question 4-30 from Fundamentals of Physics 10th Extended c2014 ed. by Halliday, Resnick and Walker

clear all
close all
clc

v_0 = 19.5;
theta_0 = 45.0;
x_player = 55.0;
g = 9.80;
t = (2*v_0*sind(theta_0))/g
x_ball = v_0*cosd(theta_0)*t
delta_x = x_ball - x_player
v_avg = delta_x/t
```

The outputs are as follows.

```
t =
    2.8140
x_ball =
   38.8010
delta_x =
  -16.1990
v_avg =
   -5.7566
```

- A lowly high diver pushes off horizontally with a speed of $2.00\,\mathrm{m\,s^{-1}}$ from the platform edge $10.0\,\mathrm{m}$ above the surface of the water. (a) At what horizontal distance from the edge is the diver $0.800\,\mathrm{s}$ 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?

```
%% Reference: Question 4-37 from Fundamentals of Physics 10th Extended c2014 ed. by Halliday, Resnick and Walker

clear all
close all
clc

v_0 = 2.0;
x_0 = 0.0;
y_0 = 10.0;
```

```
t = 0.8;
g = 9.8;
x = x_0 + (v_0 * t)
y = y_0 - (1/2)*g*t*t
t_new = sqrt((2*y_0)/g)
R = v_0 * t_new
```

The outputs are as follows.

```
x =
    1.6000
y =
    6.8640
t_new =
    1.4286
R =
    2.8571
```