**Exercise Solutions**

**Exercise 2.1**

1) >> A = [5:2:17]

A =

5 7 9 11 13 15 17

2) >> B = [3:-1:-3]'

B =

3

2

1

0

-1

-2

-3

3) >> C = A([1 4 6])

C =

5 11 15

4) >> B([3 5 7]) = C

B =

3

2

5

0

11

-2

15

**Exercise 2.2**

1)>> 1/(2+3^4) + 5/(6\*7) + 7/8

ans =

1.0061

2) >> A = linspace( 0, 2\*pi, 9 )

A =

0 0.7854 1.5708 2.3562 3.1416 3.9270 … 4.7124 5.4978 6.2832

>> SinAng = sin(A)

SinAng =

0 0.7071 1.0000 0.7071 0.0000 -0.7071 … -1.0000 -0.7071 -0.0000

3) >> SinAng = SinAng + 1

SinAng =

1.0000 1.7071 2.0000 1.7071 1.0000 0.2929 0 0.2929 1.0000

4) >> B = [1:9]

B =

1 2 3 4 5 6 7 8 9

>> SinAng.\*B

ans =

1.0000 3.4142 6.0000 6.8284 5.0000 1.7574 0 2.3431 9.0000

**Exercise 2.3**

>> A = [1 2 3; 2 3 -1; 4 -1 2]

A =

1 2 3

2 3 -1

4 -1 2

>> B = [13;4;13]

B =

13

4

13

>> X = A\B

X =

2.0000

1.0000

3.0000

**Exercise 2.4**

1) >> A = [9 12 13 0;10 3 1 5;2 5 10 3]

A =

9 12 13 0

10 3 1 5

2 5 10 3

>> B = [1 4 2 11;9 8 16 7;12 5 0 3]

B =

1 4 2 11

9 8 16 7

12 5 0 3

2) >> C = A(3,3)

C =

10

>> D = A(:,3)

D =

13

1

10

>> E = [B(1,:);B(3,:)]

E =

1 4 2 11

12 5 0 3

>> F = [A;B]

F =

9 12 13 0

10 3 1 5

2 5 10 3

1 4 2 11

9 8 16 7

12 5 0 3

>> G = [A(:,1) B(:,4)]

G =

9 11

10 7

2 3

3) >> E(2,2) = 20

E =

1 4 2 11

12 20 0 3

>> A(1,:) = 0

A =

0 0 0 0

10 3 1 5

2 5 10 3

>> F(:,3) = 1:6

F =

9 12 1 0

10 3 2 5

2 5 3 3

1 4 4 11

9 8 5 7

12 5 6 3

>> A(:,1) = B(:,2)

A =

4 0 0 0

8 3 1 5

5 5 10 3

**Exercise 4.1**

FreeFall.m

%Program to find distance travelled by freely falling object due to gravity

%at specified time intervals between given start and end times

% Input data

g = 9.81;

StartTime = input('Input the starting time ');

EndTime = input('Input the ending time ');

Increment = input('Input the time increment ');

% Calculate distance travelled at each time increment

% Create vector of times

time = StartTime:Increment:EndTime;

%Calculate the distances

distance = 1/2 \* g \* time.^2;

%% Output distances

disp('Distance Travelled in Free Fall');

disp(distance');

%% Extra task

%Create matrix of output data

table = [time;distance]';

% Output the results as a table

disp('Distance Travelled in Free Fall');

disp('Time,s Distance, m');

disp(table);

**Exercise 5.1**

1) PlotPolynomial.m

% Script to plot a polynomial

% Setup x values and calculate polynomial for those values

x = [-10:0.1:10];

y = 3\*x.^2 + 4\*x + 5;

%% Plot with a blue line

plot(x, y, 'b--');

3) FreeFallPlot.m

%Program to find distance travelled by freely falling object due to gravity

%at specified time intervals between given start and end times

% Input data

g = 9.81;

StartTime = input('Input the starting time ');

EndTime = input('Input the ending time ');

Increment = input('Input the time increment ');

% Calculate distance travelled at each time increment

% Create vector of times

time = StartTime:Increment:EndTime;

%Calculate the distances

distance = 1/2 \* g \* time.^2;

%Create matrix of output data

table = [time;distance]';

% Output the results as a table

disp('Distance Travelled in Free Fall');

disp('Time,s Distance, m');

disp(table);

%% Plot distance against time

plot(time, distance,'xr');

%% Plot distance against time with 10% margin around plot

xrange = time(end) - time(1);

yrange = distance(end) - distance(1);

xmin = time(1) - 0.1\*xrange;

xmax = time(end) + 0.1\*xrange;

ymin = distance(1) - 0.1\*yrange;

ymax = distance(end) + 0.1\*yrange;

plot(time, distance,'xr');

axis([xmin, xmax, ymin, ymax]);

**Exercise 5.2**

Projectiles.m

%Program to plot the trajectories of a projectile launched at a given

%initial velocity and launch angle

% Input data

g = 9.81; % acceleration due to gravity

StartVel = 60; % m/s

Angle = pi/3; % 60 degrees

% Create vector of times with increment 0.001s

Time = 0:0.001:15;

% Calculate horizontal displacement for each time increment

HorzDist = cos(Angle) \* (Time .\* StartVel);

% Calculate vertical displacement for each time increment

VertDist = sin(Angle) \* (Time .\* StartVel) - 0.5 \* g \* Time.^2;

%% Plot trajectory - horizontal against vertical distance

plot(HorzDist(1:end),VertDist(1:end), 'b');

% Add axis labels

xlabel('Horizontal distance, m');

ylabel('Vertical distance, m');

% Create string for title

strTitle = ['Trajectory of projectile launched at ' num2str(StartVel) ' m/s '...

'at an angle of ' num2str(Angle\*180/pi) ' degrees'];

title( strTitle );

ProjectilesExtraTask.m

%Program to plot the trajectories of two projectiles launched at the same

%initial velocities but with different launch angles

% Input data

g = 9.81; % acceleration due to gravity

StartVel = 60; % m/s

Angles = [pi/6 ; pi/3]; % 30 and 60 degrees

% Create vector of times with increment 0.001s

Time = 0:0.001:15;

% Calculate horizontal displacement for each angle

HorzDist = cos(Angles) \* (Time .\* StartVel);

% Calculate vertical displacement for each angle

VertDist = sin(Angles) \* (Time .\* StartVel);

TimeSquared = 0.5 \* g \* Time.^2;

VertDist(1,:) = VertDist(1,:) - TimeSquared;

VertDist(2,:) = VertDist(2,:) - TimeSquared;

%% Plot trajectories on same plot

plot(HorzDist(1,:),VertDist(1,:), 'b');

hold on

plot(HorzDist(2,:),VertDist(2,:), 'r');

% or

%plot(HorzDist(1,:),VertDist(1,:), 'r',HorzDist(2,:),VertDist(2,:), 'b');

%% Add axis labels

xlabel('Horizontal distance, m');

ylabel('Vertical distance, m');

% Create string for title

strTitle = ['Trajectories of projectiles launched at ' num2str(StartVel) ' m/s'];

title( strTitle );

% Add legend

strAnglesDeg = num2str(Angles \* 180 / pi);

legend(strcat(strAnglesDeg(1,:),' degrees'), strcat(strAnglesDeg(2,:), ' degrees') );

**Exercise 8.1**

1) >> A = [9 12 18 0;10 3 1 7;2 5 14 22]

A =

9 12 18 0

10 3 1 7

2 5 14 22

>> B = A >=7 & A < 15

B =

1 1 0 0

1 0 0 1

0 0 1 0

2) >> A(B) = A(B).^2

A =

81 144 18 0

100 3 1 49

2 5 196 22

3) >> find(A>100)

ans =

4

9

4) >> A(A>100)

ans =

144

196

**Exercise 8.2**

ProjectilesLimitCheck.m

%Program to plot the trajectories of a projectile launched at a given

%initial velocity and launch angle

%% Input data

g = 9.81; % acceleration due to gravity

StartVel = 60; % m/s

Angle = input('Input angle in degrees ');

if ( Angle < 0 || Angle > 90 )

disp('Angle is out of range');

% Calculate distances

% Convert angle to radians

Angle = Angle \* pi /180;

% Create vector of times with increment 0.001s

Time = 0:0.001:15;

% Calculate horizontal displacement for each time increment

HorzDist = cos(Angle) \* (Time .\* StartVel);

% Calculate vertical displacement for each time increment

VertDist = sin(Angle) \* (Time .\* StartVel) - 0.5 \* g \* Time.^2;

% Plot only positive vertical displacements

% Find index of first negative vertical distance

index = find(VertDist < 0, 1 );

% Plot trajectory - horizontal against vertical distance

plot(HorzDist(1:index-1),VertDist(1:index-1), 'b');

%% Add axis labels

xlabel('Horizontal distance, m');

ylabel('Vertical distance, m');

% Create string for title

strTitle = ['Trajectory of projectile launched at ' num2str(StartVel) ' m/s '...

'at an angle of ' num2str(Angle\*180/pi) ' degrees'];

title( strTitle );

end

**Exercise 8.3**

% Assign class of degree based on mark

mark = input('Please input mark: ');

if ( mark >= 0 && mark <=100 )

if mark >= 70

disp('First class degree');

elseif mark >= 60

disp('Upper second class degree');

elseif mark >= 50

disp('Lower second class degree');

elseif mark >= 40

disp('Third class degree');

else

disp('Fail - no degree awarded');

end

else

disp('Invalid mark');

end

**Exercise 9.1**

FarenheitToCelcius.m

%% Conversion of temperatures from Farenheit to Celcius

%% Set up vector of farenheit values

Tf = 0:1000;

%% Convert to Celcius using for loop without preallocation

tic

ind = 1;

for T = Tf

Tc(ind) = (T - 32) \* 5/9;

ind = ind +1;

end

toc;

%% Loop with preallocated vector

tic

Tc = zeros(size(Tf));

ind = 1;

for T = Tf

Tc(ind) = (T - 32) \* 5/9;

ind = ind +1;

end

toc;

%% Vectorised

Tc = [];

tic

Tc = (Tf - 32) \* 5/9;

toc

**Exercise 9.2**

ProjectilesLoopInput.m

%Program to plot the trajectories of two projectiles launched at the same

%initial velocities but with different launch angles

% Input data

g = 9.81; % acceleration due to gravity

StartVel = 60; % m/s

% Loop until valid values of the two angles have been input

Angles = -1;

while ( Angles < 0 || Angles > 90 )

Angles = input('Input first angle in degrees ');

end

Angles(2,1) = -1;

while ( Angles(2,1) < 0 || Angles(2,1) > 90 )

Angles(2,1) = input('Input second angle ');

end

Angles = Angles \* pi /180;

% Create vector of times with increment 0.001s

Time = 0:0.001:15;

% Calculate horizontal displacement for each angle

HorzDist = cos(Angles) \* (Time .\* StartVel);

% Calculate vertical displacement for each angle

VertDist = sin(Angles) \* (Time .\* StartVel);

TimeSquared = 0.5 \* g \* Time.^2;

VertDist(1,:) = VertDist(1,:) - TimeSquared;

VertDist(2,:) = VertDist(2,:) - TimeSquared;

% Plot trajectories on same plot

index = find(VertDist(1,:) >= 0 );

plot(HorzDist(1,index),VertDist(1,index), 'b');

hold on

index = [];

index = find(VertDist(2,:) >= 0 );

plot(HorzDist(2,index),VertDist(2,index), 'r');

% or

%plot(HorzDist(1,:),VertDist(1,:), 'r',HorzDist(2,:),VertDist(2,:), 'b');

% Add axis labels

xlabel('Horizontal distance, m');

ylabel('Vertical distance, m');

% Create string for title

strTitle = sprintf('Trajectories of projectiles launched at %4.2f m/s',StartVel);

title( strTitle );

% Add legend

strAnglesDeg = num2str(Angles \* 180 / pi);

legend(strcat(strAnglesDeg(1,:),' degrees'), strcat(strAnglesDeg(2,:), ' degrees') );

**Exercise 10.1**

CelsiusToFahrenheit.m

function [ DegreesF ] = CelsiusToFahrenheit( DegreesC )

%Function to convert degrees Celsius to Fahrenheit

% input - temp in Celsius

% output - temp in Fahrenheit

DegreesF = DegreesC\*9/5 + 32;

end

TestCelsiusToFahrenheit.m

% Test program for CelsiusToFahrenheit script

% Test for two known values: 0 and 100 degrees C

tempC = 0;

tempF = CelsiusToFahrenheit(tempC);

disp([num2str(tempC) ' degrees C = ' num2str(tempF) ' degrees F']);

tempC = 100;

tempF = CelsiusToFahrenheit(tempC);

disp([num2str(tempC) ' degrees C = ' num2str(tempF) ' degrees F']);

% Test with a vector of values

tempC = [0 23 45 100];

tempF = CelsiusToFahrenheit(tempC);

results = [ tempC' tempF'];

disp(['degrees C degrees F']);

disp(results);

CalcPolynomial.m

function [y] = CalcPolynomial( x )

% Function to calculate a polynomial

y = 3\*x.^2 + 4\*x + 5;

TestPolynomial.m

% Test program for CalcPlotPolynomial

% Test for two known values: 0 and 100 degrees C

x = 0;

y = CalcPolynomial(x);

disp([' x = ' num2str(x) ' y = ' num2str(y)]);

x = 10;

y = CalcPolynomial(x);

disp([' x = ' num2str(x) ' y = ' num2str(y)]);

% Test with a vector of values

x = [0:5];

y = CalcPolynomial(x);

results = [ x' y'];

disp(['x y']);

disp(results);

**Exercise 10.2**

>> poly1 = @(x) 3.\*x.^2 + 4.\*x +5;

>> plot([-10:10], poly1(-10:10) )

>> fplot(poly1, [0,20])

>> poly(1:10)

ans =

12 25 44 69 100 137 180 229 284 345

**Exercise 11.1**

SmokerWeightHeightPlot.m

load patients

% Create table using selected fields

PatientData = table(LastName,Smoker,Weight,Height);

% Plot weight vs height for smokers

plot(PatientData.Weight(Smoker), PatientData.Height(Smoker), 'xb');

hold on

% Invert the selection to plot the data for non-smokers

plot(PatientData.Weight(~Smoker), PatientData.Height(~Smoker), 'or');

xlabel('Weight (lb)');

ylabel('Height (in)');

legend('Smoker','Non-smoker');

**Exercise 12.1**

ImportMedData.m

NB. This file is auto-generated by the Import Wizard

function [Age1,BMI1] = ImportMedData(filename, startRow, endRow)

%IMPORTFILE1 Import numeric data from a text file as column vectors.

% [AGE1,BMI1] = IMPORTFILE1(FILENAME) Reads data from text file FILENAME

% for the default selection.

%

% [AGE1,BMI1] = IMPORTFILE1(FILENAME, STARTROW, ENDROW) Reads data from

% rows STARTROW through ENDROW of text file FILENAME.

%

% Example:

% [Age1,BMI1] = importfile1('MedicalData.txt',2, 5611);

%

% See also TEXTSCAN.

% Auto-generated by MATLAB on 2017/04/04 14:48:42

%% Initialize variables.

delimiter = '\t';

if nargin<=2

startRow = 2;

endRow = inf;

end

%% Format string for each line of text:

% column3: double (%f)

% column10: double (%f)

% For more information, see the TEXTSCAN documentation.

formatSpec = '%\*s%\*s%f%\*s%\*s%\*s%\*s%\*s%\*s%f%\*s%\*s%\*s%\*s%\*s%\*s%\*s%\*s%\*s%\*s%\*s%[^\n\r]';

%% Open the text file.

fileID = fopen(filename,'r');

%% Read columns of data according to format string.

% This call is based on the structure of the file used to generate this

% code. If an error occurs for a different file, try regenerating the code

% from the Import Tool.

dataArray = textscan(fileID, formatSpec, endRow(1)-startRow(1)+1, 'Delimiter', delimiter, 'EmptyValue' ,NaN,'HeaderLines', startRow(1)-1, 'ReturnOnError', false);

for block=2:length(startRow)

frewind(fileID);

dataArrayBlock = textscan(fileID, formatSpec, endRow(block)-startRow(block)+1, 'Delimiter', delimiter, 'EmptyValue' ,NaN,'HeaderLines', startRow(block)-1, 'ReturnOnError', false);

for col=1:length(dataArray)

dataArray{col} = [dataArray{col};dataArrayBlock{col}];

end

end

%% Close the text file.

fclose(fileID);

%% Post processing for unimportable data.

% No unimportable data rules were applied during the import, so no post

% processing code is included. To generate code which works for

% unimportable data, select unimportable cells in a file and regenerate the

% script.

%% Allocate imported array to column variable names

Age1 = dataArray{:, 1};

BMI1 = dataArray{:, 2};

PlotBMI.m

% Import and plot age and BMI data

% Use the function generated by the Import Wizard to load the data

[Age,BMI] = ImportMedData('MedicalData.txt',2,1001);

% Clean the data

badData = isnan(Age)|isnan(BMI);

CleanAge = Age(~badData);

CleanBMI = BMI(~badData);

% Plot as a scatter graph

plot(Age/12,BMI,'o');

xlabel('Age, years');

ylabel('BMI');

title('Age vs BMI');