Homework #5

Table of Contents

Ian Blackstone, Helena-Nikolai Fujishin
Math 365, Fall 2016

Problems

```
function hmwk1()
hmwk_problem(@prob1,'prob1');
hmwk_problem(@prob2,'prob2');
hmwk_problem(@prob3,'prob3');
hmwk_problem(@prob4,'prob4');
hmwk_problem(@prob5,'prob5');
end
function hmwk_problem(prob,msg)
try
    prob()
    fprintf('%s : Success!\n',msg);
catch me
    fprintf('%s : Something went wrong.\n',msg);
    fprintf('%s\n',me.message);
end
fprintf('\n');
end
```

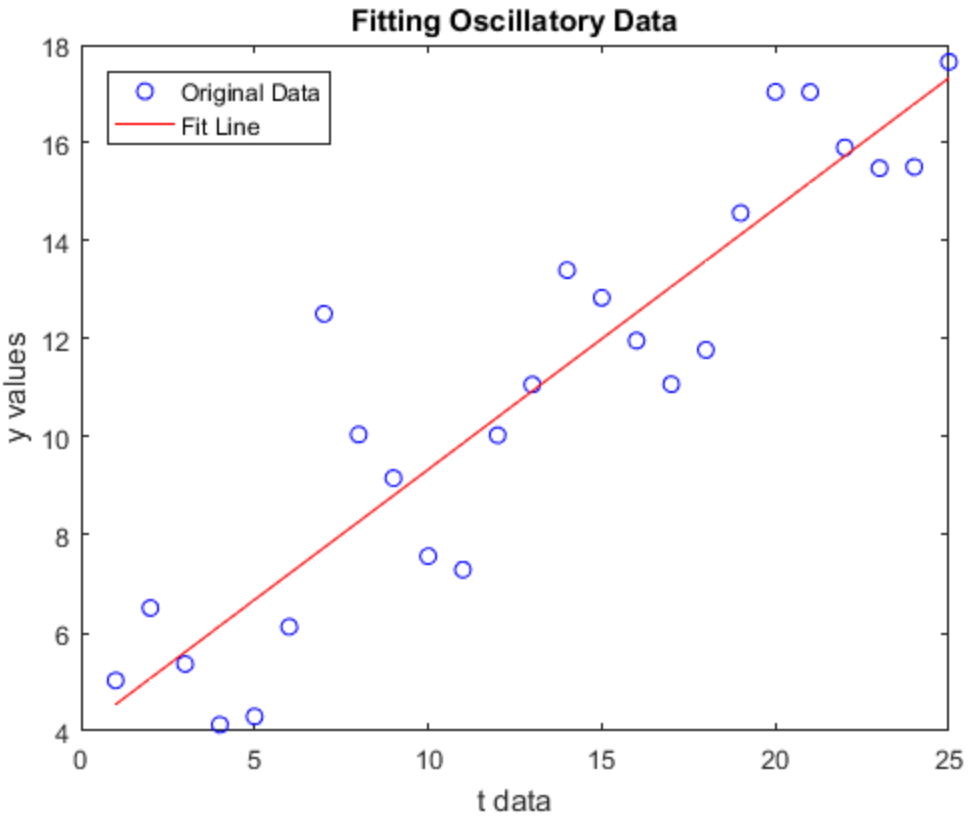
Problem #1 : Oscillatory Data Fitting NCM 5.8

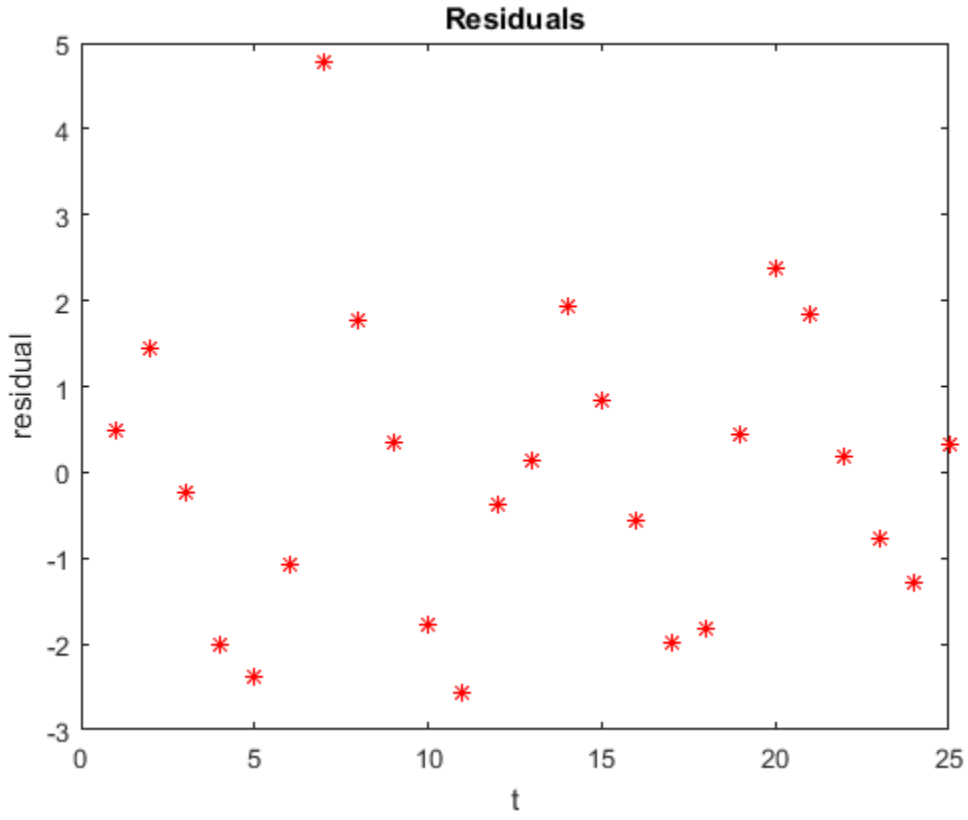
```
function prob1()
% In the problem we were given the data:
tdata = [1:25];
ydata = [5.0291 6.5099 5.3666 4.1272 4.2948 6.1261 12.5140 10.0502
 9.1614 7.5677 7.2920 10.0357 11.0708 13.4045 12.8415 11.9666 11.0765
 11.7774 14.5701 17.0440 17.0398 15.9069 15.4850 15.5112 17.6572];
y1 = ydata';
% We know that the system is set up like y=Bt where B is a column
 vector of
% our coeffiencts Beta(1) and Beta(2) to fit the linear equation y=
 Beta(1)+Beta(2)*t

% PART A
```

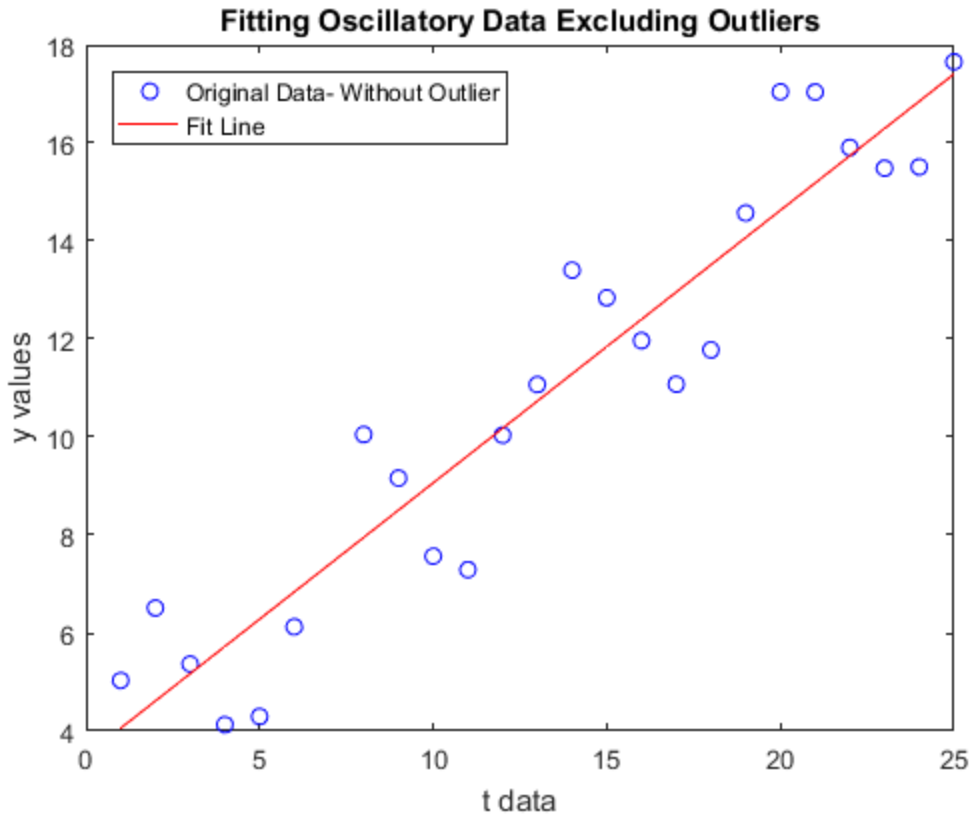
```
t1 = ones(length(tdata),1); % We create a new vector to be our t for
 the y=Bt
% by creating a column of ones...
tnew = [t1 , tdata']; %...and then setting it so that we have a column
 of
% ones next to a column made of tdata
B = tnew\y1; % To solve for our coefficients we use the backslash
 operator
y2 = B(1)+(B(2).*tdata); % And plug it into our equation setting the
 new equation
figure
plot(tdata,ydata,'bo',tdata,y2,'r-');% We plot our data points and our
 given line
title('Fitting Oscillatory Data')
ylabel('y values')
xlabel('t data')
legend('Original Data','Fit Line','Location','northwest')

% Need to plot residuals
figure
plot(tdata,ydata-y2,'r*')
title('Residuals')
ylabel('residual')
xlabel('t')

% PART B
% We discard the outlier by hand, note changes to some variables:
tdata2 = tdata;
ydata2 = ydata;
ydata2(7) = [];
tdata2(7) = [];
y1 = ydata2';
t1 = ones(length(tdata2),1);
tnew = [t1 , tdata2'];
B = tnew\y1;
y2new = B(1)+(B(2).*tdata2);

% We plot the residuals again and note the differences:
figure
plot(tdata2,ydata2,'bo',tdata2,y2new,'r-');%We plot our data points
 and our given line without outlier
title('Fitting Oscillatory Data Excluding Outliers')
ylabel('y values')
xlabel('t data')
legend('Original Data- Without Outlier','Fit
 Line','Location','northwest')

% Need to plot residuals
figure
plot(tdata2,ydata2-y2new,'r*')
title('Residuals Excluding Outlier')
ylabel('residual')
xlabel('t')
```

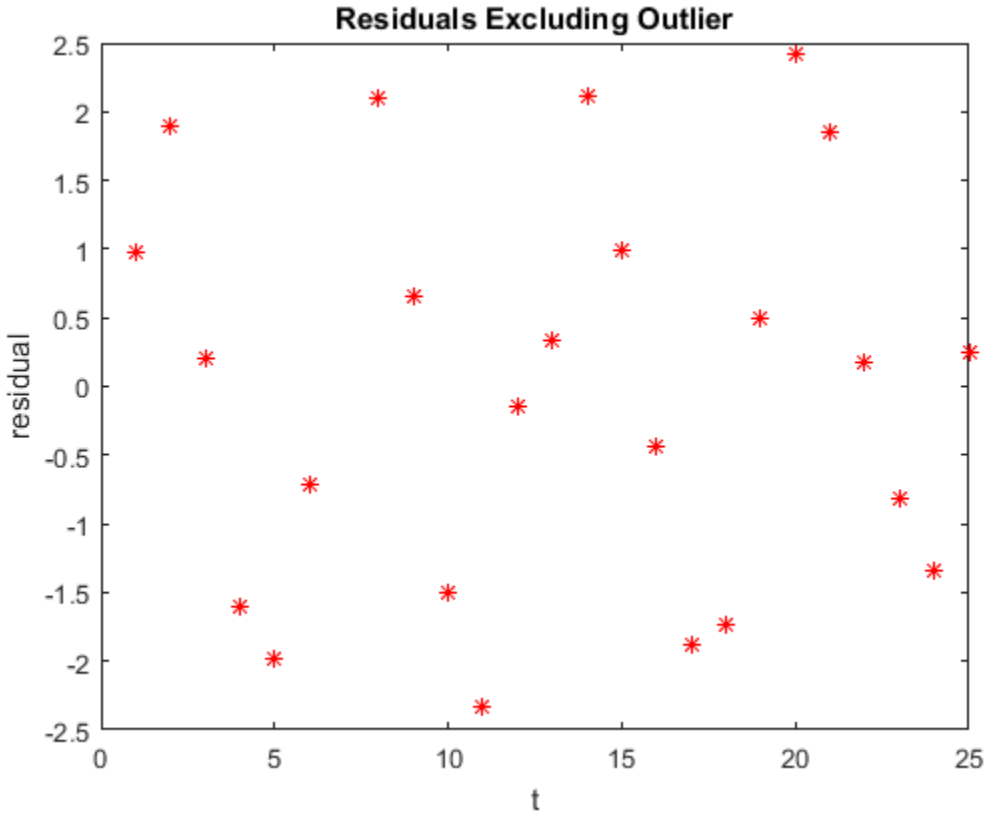
```
% PART C
% We fit the data, excluding the outlier, to
% y(t)=Beta(1)+Beta(2)t+Beta(3)sin(t)
tnow = [ones(length(tdata2),1), tdata2', sin(tdata2')]; %We make our
 new matrix without the outlier for t
B = tnow\y1;
y3 = B(1)+(B(2).*tnow)+(B(3).*sin(tnow));
figure
plot(tdata2,ydata2,'bo',tnow,y3,'r-');%We plot our data points and our
 given line without outlier
title('Fitting Oscillatory Data Excluding Outliers')
ylabel('y values')
xlabel('t data')
legend('Original Data- Without Outlier','Fit
 Line','Location','southeast')

% PART D
% We evaluate our equation above over [0,26], including the outlier
 marked
% with *
tt = linspace(0,26,100);
yy = B(1)+(B(2).*tt)+(B(3).*sin(tt));
% The question says to plot the third fit on a finer grid, but it
 doesn't
% say whether or not to include the outliner, only to mark it so that
 is
% what I have done:
figure
plot(tdata(7),ydata(7),'b*',tdata2,ydata2,'o',tt,yy,'-')
title('Best fit line for the data with outliers highlighted')
ylabel('y')
xlabel('t')
legend('outlier','data excluding outlier','best fit line without
 outlier','Location','southeast')
end
```

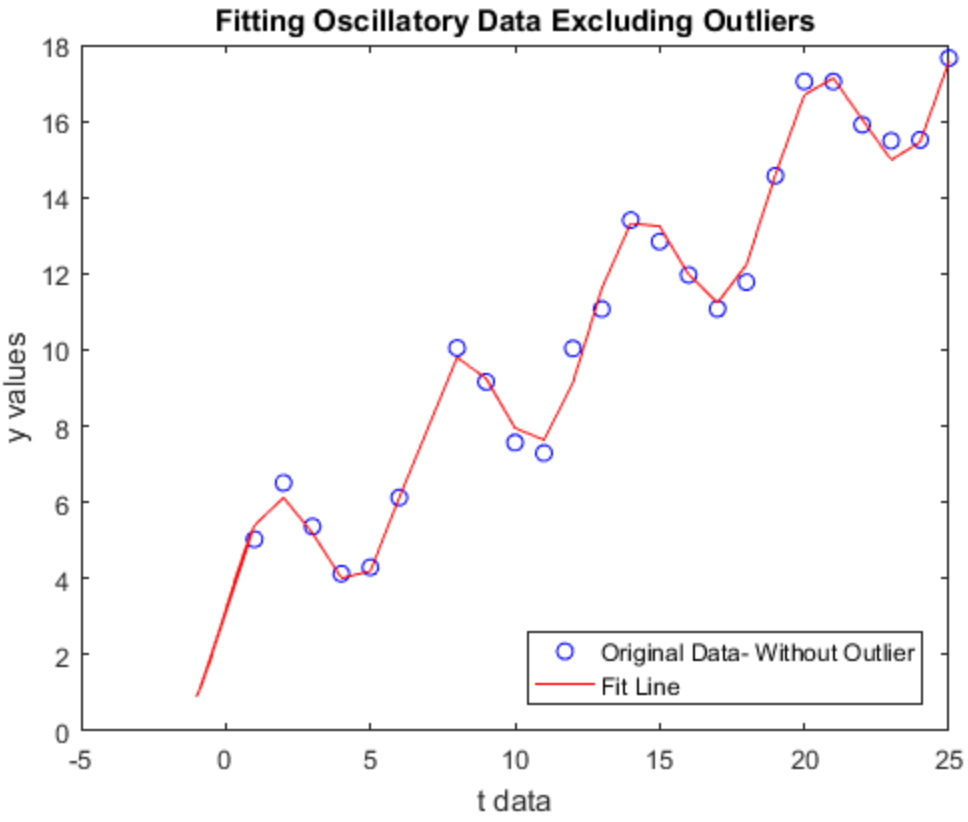
```
prob1 : Success!
```

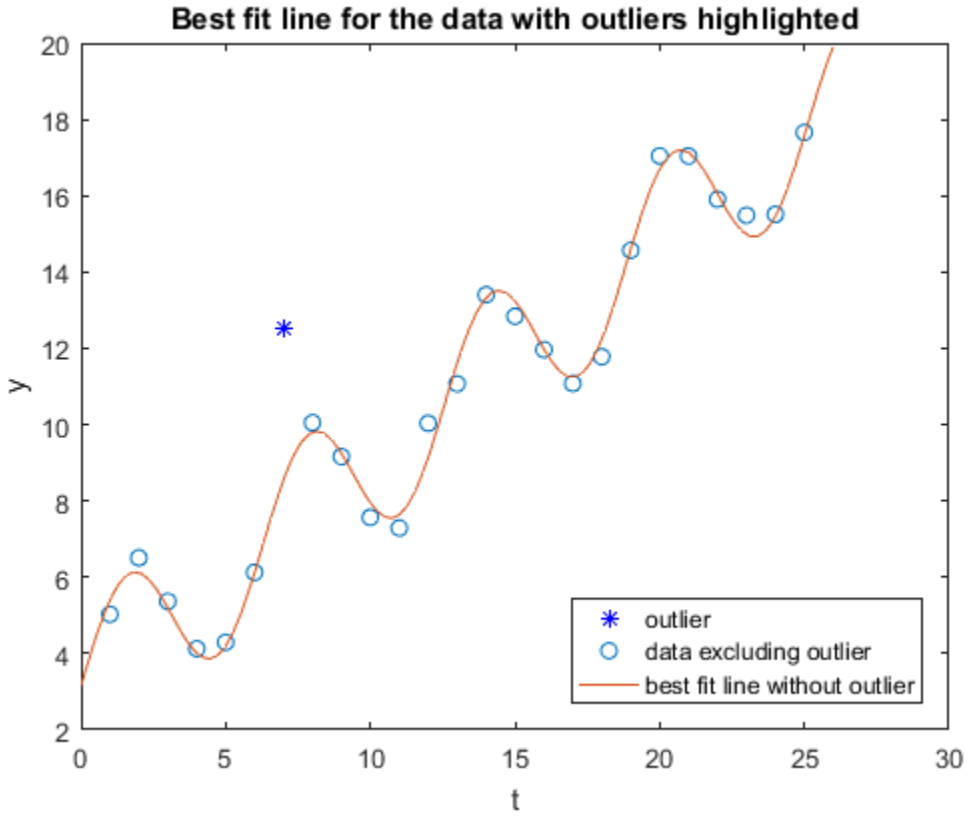

Fitting Oscillatory Data
Original Data
Fit Line
y values
t data
18
16
14
12
10
8
6
4
0
5
10
15
20
25


Residuals
residual
t
5
4
3
2
1
0
-1
-2
-3
0
5
10
15
20
25


Fitting Oscillatory Data Excluding Outliers
Original Data- Without Outlier
Fit Line
y values
t data
18
16
14
12
10
8
6
4
0
5
10
15
20
25


Residuals Excluding Outlier
residual
t
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
5
10
15
20
25


Fitting Oscillatory Data Excluding Outliers
y values
t data
Original Data- Without Outlier
Fit Line

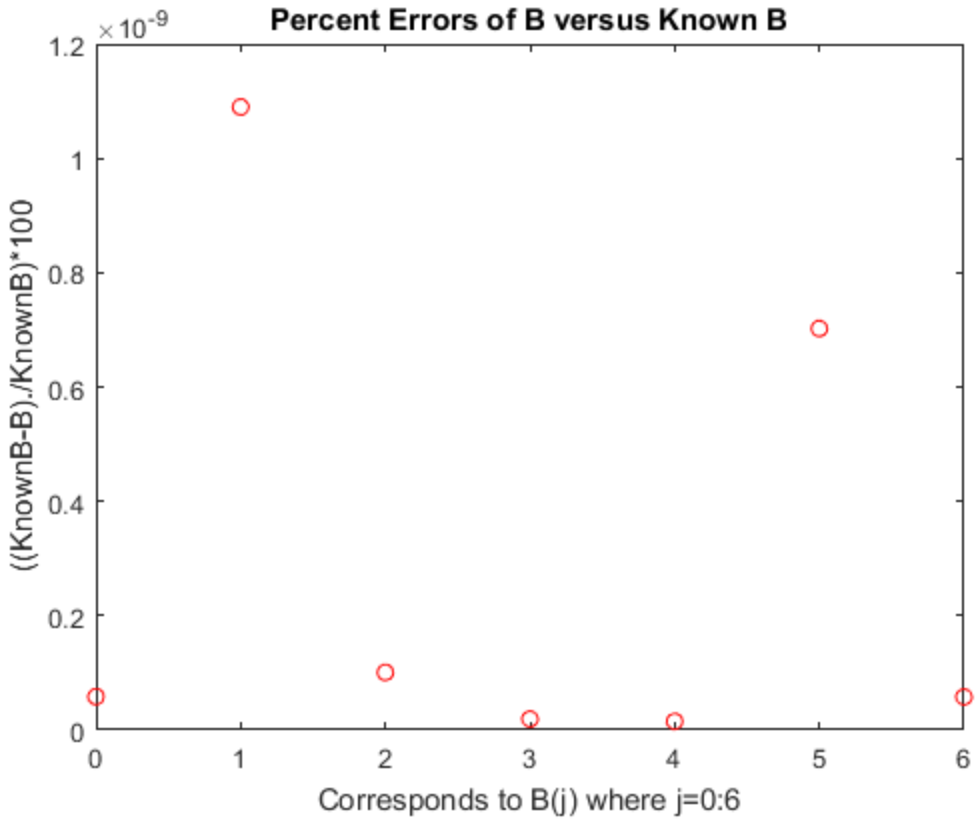

Best fit line for the data with outliers highlighted
y
t
outlier
data excluding outlier
best fit line without outlier

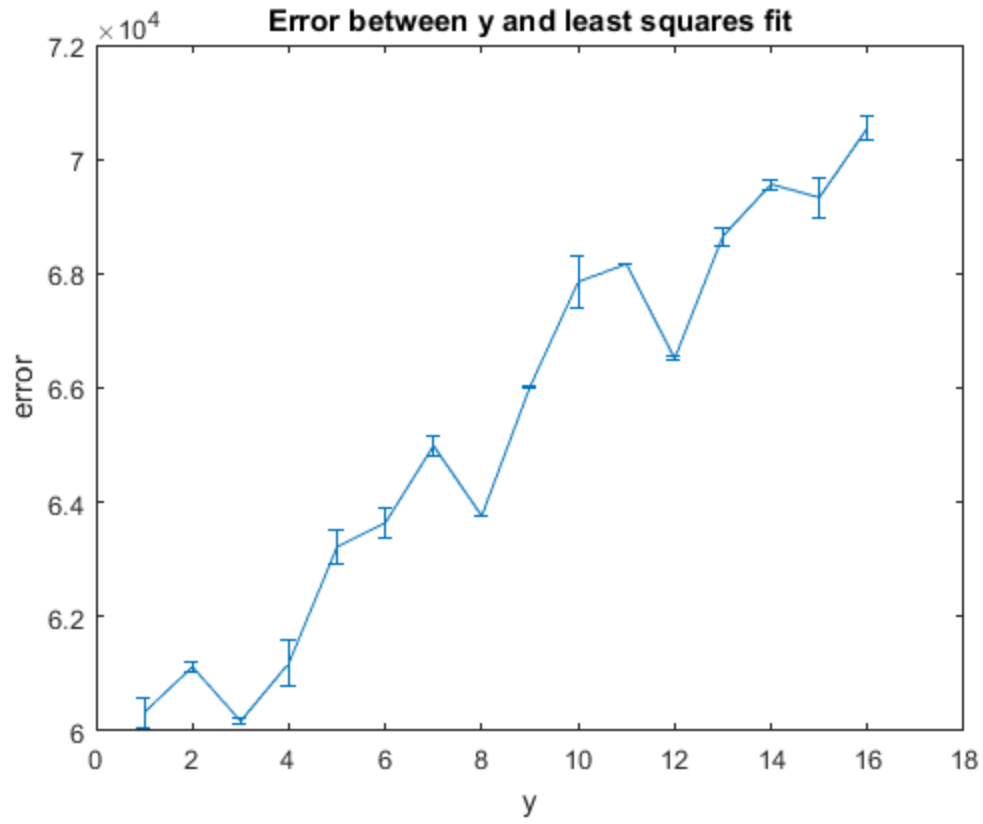
Problem #2 : Longley Data Set NCM 5.11

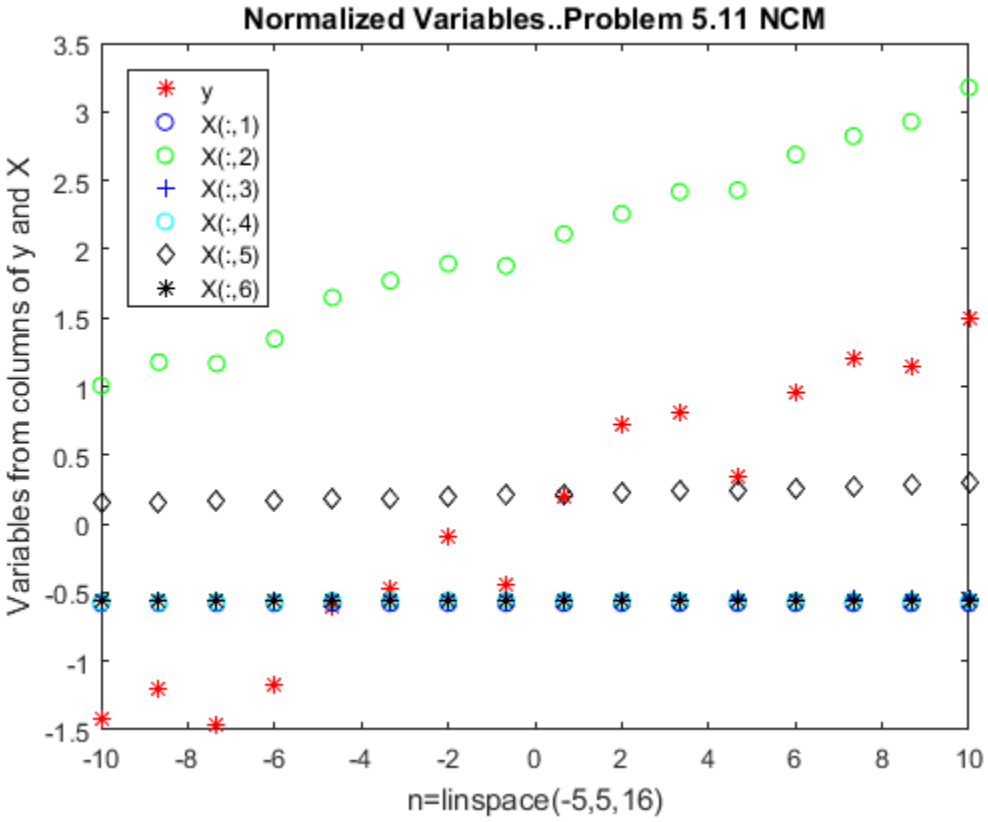
```
function prob2()
load longley.dat
y = longley(:,1);
X = longley(:,2:7);
X1 = [ones(length(X),1),X];
% PART A
% Use backslash operator to compute our Bs
B = X1\y;
% PART B
% Compare our Bs with the known Bs
% I will do this by calculating the errors:
KnownB = [-3482258.63459582; 15.0618722713733; -0.358191792925910e-01;
 -2.02022980381683;  -1.03322686717359;-0.511041056535807e-01;
 1829.15146461355];
error = ((KnownB-B)./KnownB)*100;
figure
plot(0:6,error','ro')
title('Percent Errors of B versus Known B')
xlabel('Corresponds to B(j) where j=0:6')
ylabel('((KnownB-B)./KnownB)*100')
% PART C
% Use errorbar to plot y with error bars whose magnitude is the
 difference
% between y and the least squares fit
lsf = X1*B;
figure
errorbar(y,y-lsf)
title('Error between y and least squares fit')
ylabel('error')
xlabel('y')

% PART D
% Use corrcoef to compute the correlation coefficients for X without
 the
% column of 1's. Which variables are highly correlated?
coefs = corrcoef(X);
highcorr = coefs - triu(coefs) > 0.7
% From highcorr we can see the spots of the variables that are highly
% correlated.

% PART E
% Normalize the vector y so that its mean is zero and its standard
 deviation
% is one.
y = y - mean(y(:));
y = y/std(y(:));
% Do the same thing to the columns of X. Now plot all seven normalized
% variables on the same axis. Include a legend.
X = X - mean(X(:));
X = X/std(X(:));
n = linspace(-10,10,16);
```

```
figure
plot(n,y,'r*',n,X(:,1),'bo',n,X(:,2),'go',n,X(:,3),'b
+',n,X(:,4),'co',n,X(:,5),'kd',n,X(:,6),'k*')
title('Normalized Variables..Problem 5.11 NCM')
ylabel('Variables from columns of y and X')
xlabel('n=linspace(-5,5,16)')
legend('y','X(:,1)','X(:,2)','X(:,3)','X(:,4)','X(:,5)','X(:,6)','Location','north
end
```

```
highcorr =

     0     0     0     0     0     0
     1     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     1     1     0     0     0     0
     1     1     0     0     1     0

prob2 : Success!
```




Error between y and least squares fit
×10^4
7.2
7
6.8
6.6
6.4
6.2
6
error
0
2
4
6
8
10
12
14
16
18
y

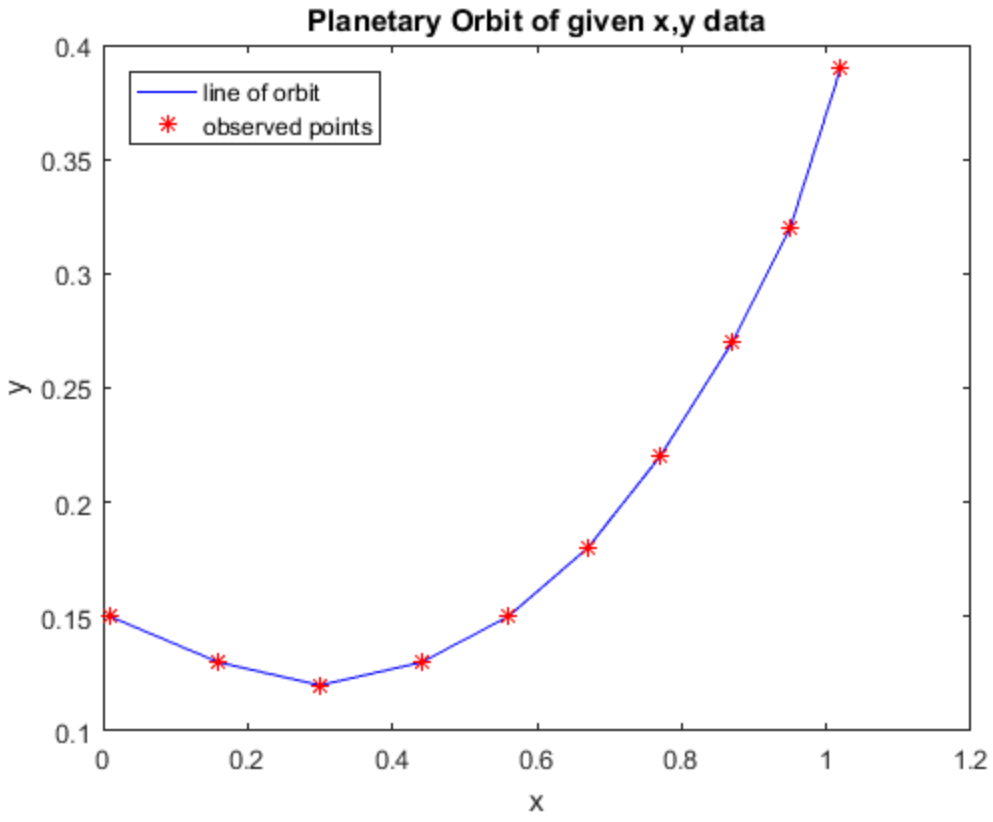

Normalized Variables..Problem 5.11 NCM
y
X(:,1)
X(:,2)
X(:,3)
X(:,4)
X(:,5)
X(:,6)
Variables from columns of y and X
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-10
-8
-6
-4
-2
0
2
4
6
8
10
n=linspace(-5,5,16)

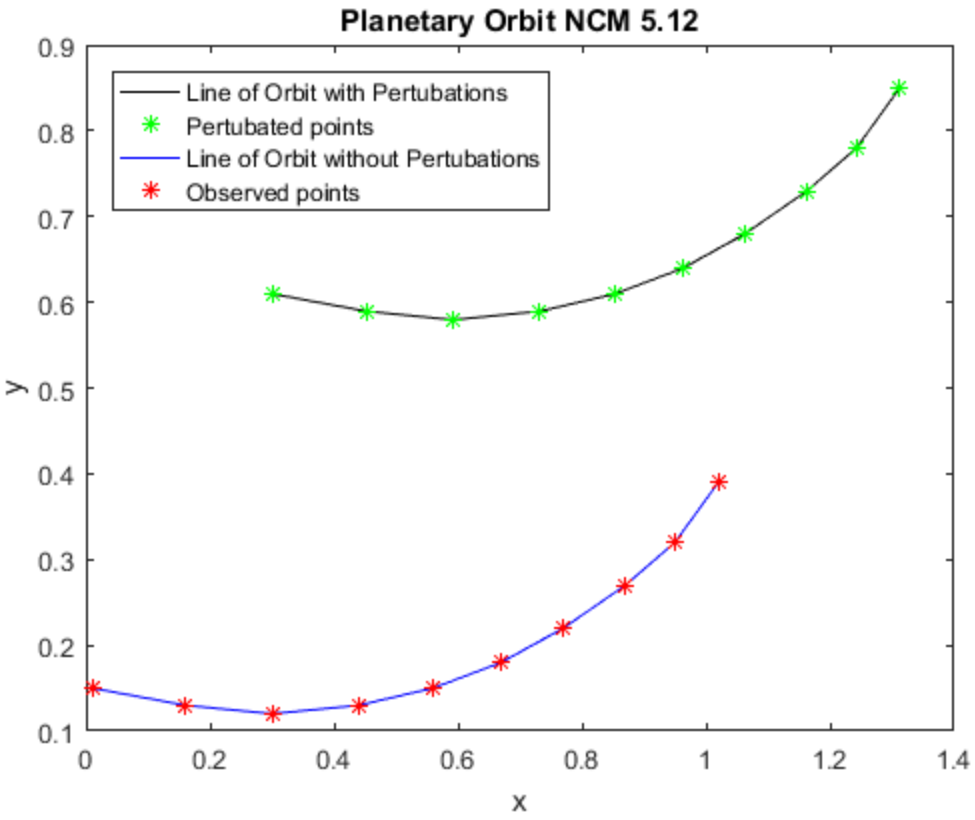
Problem #3 : Fitting Planetary Orbits NCM 5.12

```
function prob3()
x = [1.02 .95 .87 .77 .67 .56 .44 .30 .16 .01]';
y = [0.39 .32 .27 .22 .18 .15 .13 .12 .13 .15]';

%PART A
% Determine the coefficients in the quadratic form that fits these
 data in
% the least squares sense by setting one of the coefficients equal to
 one and
% solving a 10-by-5 overdetermined system of linear equations for the
 other
% five coefficients. Plot the orbit with x on the x-axis and y on the
 y-axis.
% Superimpose the ten data points on the plot.
M = [(x.^2), (x.*y) (y.*y) x y];
t = ones(10,1)*-1;
c = M\t;
figure
plot(x,y,'b-',x,y,'r*')
title('Planetary Orbit of given x,y data')
legend('line of orbit','observed points','Location','northwest')
ylabel('y')
xlabel('x')


%PART B
% (b) This least squares problem is nearly rank deficient. To see what
 effect this
% has on the solution, perturb the data slightly by adding to each
 coordinate
% of each data point a random number uniformly distributed in the
 interval
% [0.0005, .0005]. Compute the new coefficients resulting from the
 perturbed
% data. Plot the new orbit on the same plot with the old orbit.
 Comment on
% your comparison of the sets of coefficients and the orbits.
xnew = x+((rand)-(0.5));
ynew = y+((rand)-(0.5));
Mnew = [(xnew.^2), (xnew.*ynew) (ynew.*ynew) xnew ynew];
tnew = ones(10,1)*-1;
cnew = Mnew\tnew;
figure
plot(xnew,ynew,'k-',xnew,ynew,'g*',x,y,'b-',x,y,'r*')
title('Planetary Orbit NCM 5.12')
legend('Line of Orbit with Pertubations','Pertubated points','Line of
 Orbit without Pertubations','Observed points','Location','northwest')
ylabel('y')
xlabel('x')

% I would like to compare the coefficients:
```

```
Cerrorspercent = zeros(5,1);

for j = 1:5
    Cerrorspercent(j) = abs((c(j)-cnew(j))/c(j))*100;
end
CoefficientList = {'coeff1','coeff2','coeff3','coeff4','coeff5'};
T = table(Cerrorspercent,'RowNames',CoefficientList)
% From the table we see that some of Coefficients are within a
 magnitude and
% some others are very far off
end
```

```
T =

              Cerrorspercent
              ______________

    coeff1     83.832
    coeff2     1082.2
    coeff3     81.737
    coeff4     70.282
    coeff5      71.54

prob3 : Success!
```

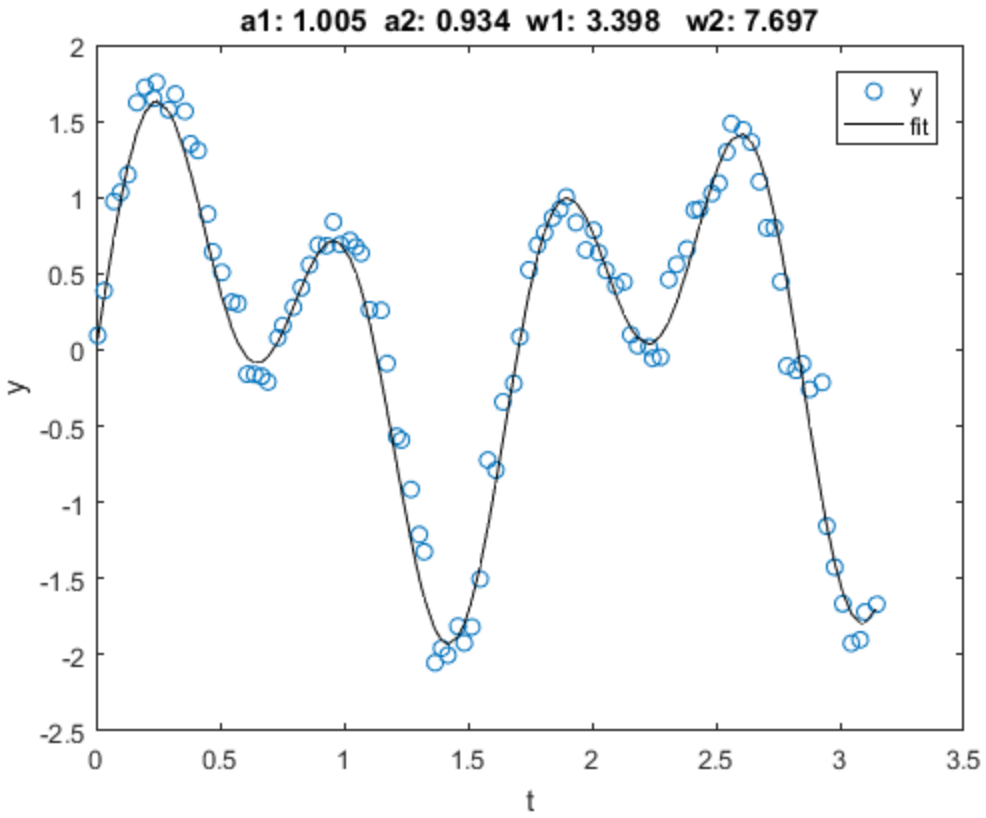




Problem #4 : Periodic Fit

```
function prob4()

% These functions are saved as external files then called.
% function sinfit

% % Load the data.
% load('PeriodicData.mat')

% % Draw a figure.
% clf
% shg
% set(gcf,'doublebuffer','on')
% h = plot(t,y,'o',t,0*t,'-');
% h(3) = title('');
% ylabel('y')
% xlabel('t')
% legend('y','fit')

% % Declare initial guess for omega.
% w0 = [3 7]';

% % Find new values for omega and amplitude by minimizing the residual
 residual.
```

```
% w = fminsearch(@findvar,w0,[],t,y,h);

% % Draw the final function.
% set(h(2),'color','black')
% end

% function res = findvar(w,t,y,h)
% % Make an array of zeros.
% m = length(t);
% n = length(w);
% X = zeros(m,n);

% % fill the array with the values of sin(w1t) and sin(w2t).
% for j = 1:n
% X(:,j) = sin(w(j)*t);
% end

% % Solve the linear equations to find the amplitude of each wave
 function.
% a = X\y;
% z = X*a;

% % find the residual.
% res = norm(z-y);

% % plot the function as it evolves.
% set(h(2),'ydata',z);
% set(h(3),'string',sprintf('a1: %.3f  a2: %.3f  w1: %.3f   w2:
 %.3f',a,w))
% pause(.1)
% end

sinfit()

end

prob4 : Success!
```



Problem #5 : Simpson's Rule

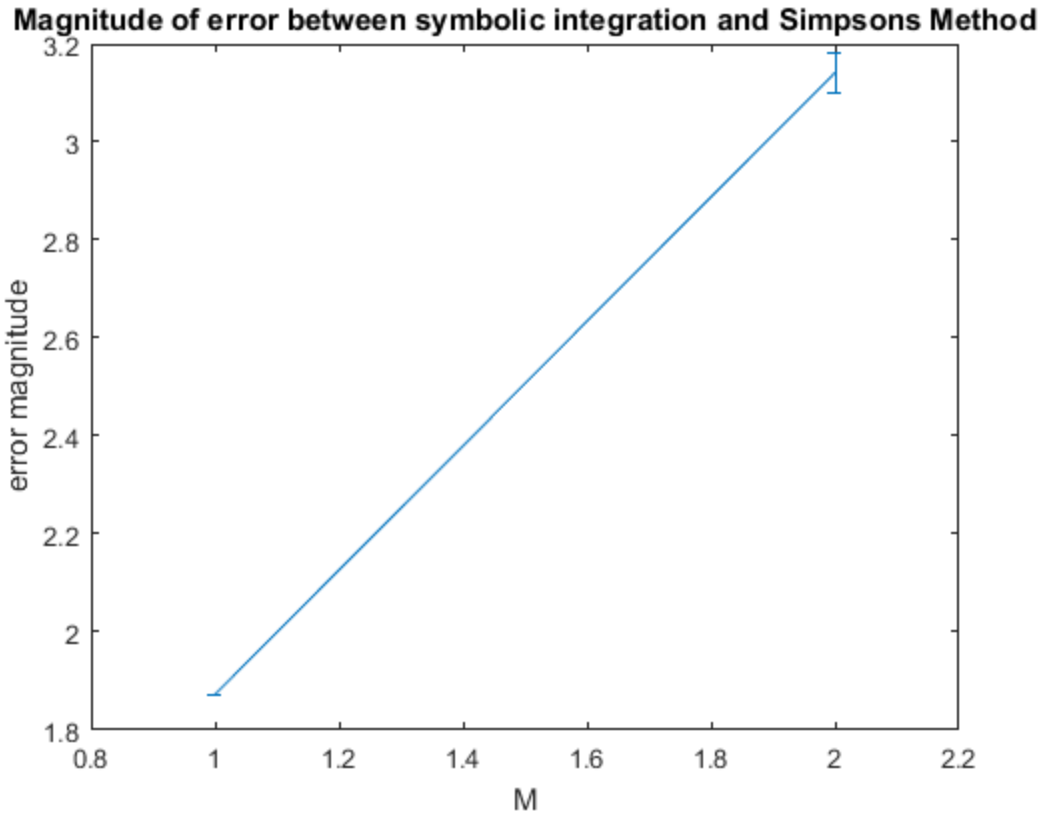
```
function prob5()

% The following function is called from an outside file.
% function x = simps(a,b,n,f)
% % The function implements the composite Simpson's rule

% h = (b-a)/n;
% x = zeros(1,n+1);
% x(1) = a;
% x(n+1) = b;
% p = 0;
% q = 0;

% % Define the x-vector
% for i = 2:n
%     x(i) = a + (i-1)*h;
% end

% % Define the terms to be multiplied by 4
% for i = 2:((n+1)/2)
%     p = p + (f(x(2*i -2)));
% end
```

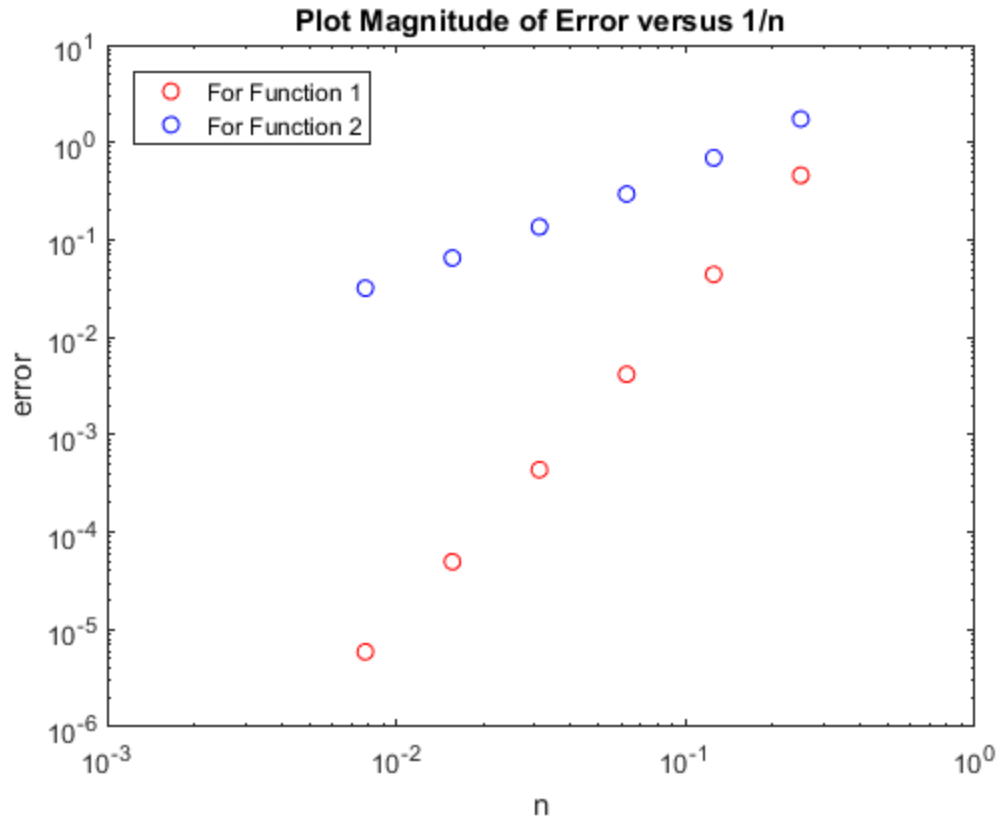
```
% % Define the terms to be multiplied by 2
% for i = 2:((n-1)/2)
%     q = q + (f(x(2*i -1)));
% end

% Calculate final output
% x = (h/3)*(f(a) + 2*q + 4*p + f(b));
% end

% PART A
% Use Matlab's symbolic tool box to obtain exact answers to I(f1) (and
 I(f2))

syms x
f1 = @(x) (-1+x)^2*exp(-(x^2));
f2 = @(x) 2*(1/(1+x^2));
I1a = int(f1,x,-1,1);
I2a = int(f2,x,-1,1);

% PART B
% Write Matlab Function for approximating a general integral.
I1b = simps(-1,1,100,f1); %I am just using 100 here for n as a general
 value
I2b = simps(-1,1,100,f2);
% Let us see how they compare, for funsies:
M = [I1a;I2a];
ERROR = [(I1a-I1b); (I2a-I2b)];
figure
errorbar(M,ERROR)
title('Magnitude of error between symbolic integration and Simpsons
 Method')
ylabel('error magnitude')
xlabel('M')

% PART C
% Using your Simpson's rule method from part (a), compute an
 approximation to I(f1) and
% I(f2) for n = 4, 8, 16, 32, 64, 128.
% I could do a for loop on all these but I want to see how it all
 plays out:
Ns = {'f1 n=4';'f1 n=8';'f1 n=16';'f1 n=32';'f2 n=64';'f2 n=128'};
n = [4 8 16 32 64 128];
I1b = ones(6,1);
I2b = ones(6,1);

% Let's make a loop to calculate each integration:
for j = 1:length(n)
    I1b(j) = simps(-1,1,n(j),f1);
    I2b(j) = simps(-1,1,n(j),f2);
end

% Report the magnitude of the error in these approximations for each n
 in a nice table.
I1berror = ones(length(n),1);
```

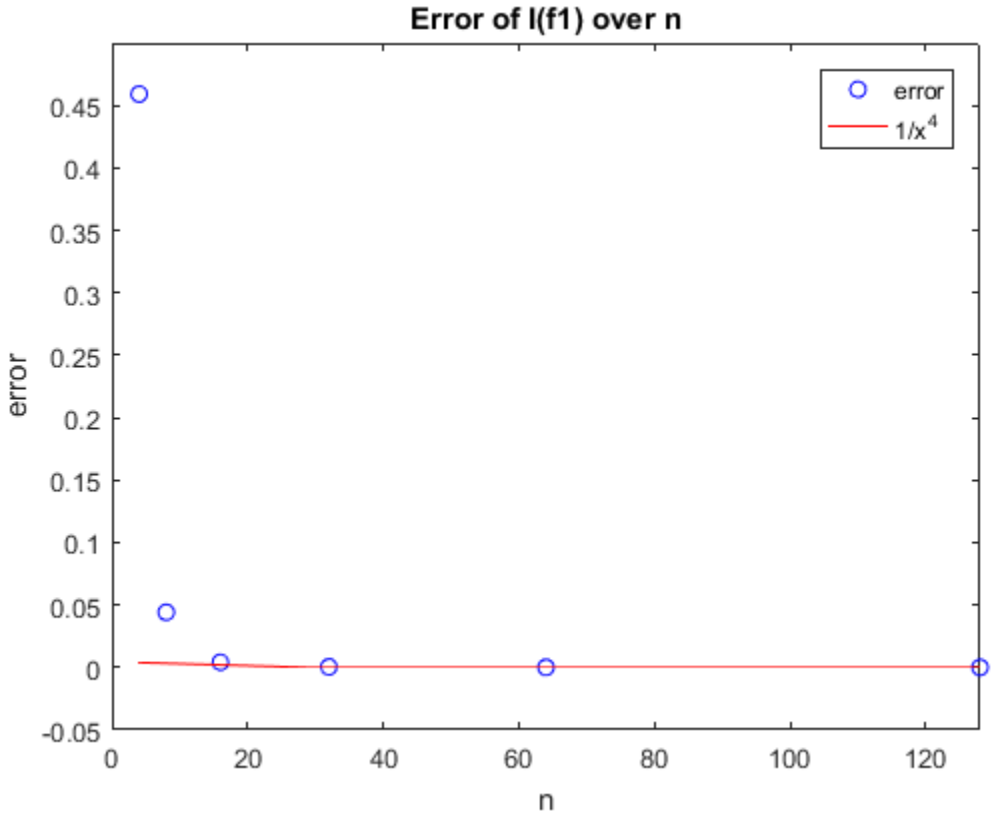
```
I2berror = ones(length(n),1);
for j = 1:length(n)
    I1berror(j) = I1a-I1b(j);
    I2berror(j) = I2a-I2b(j);
end

% Table of errors
T = table(I1berror,I2berror,'RowNames',Ns)

%Produce a plot of the magnitude of the error vs. 1/n (on a log-log
 scale).
n = [4 8 16 32 64 128];
nnew = 1./n;
figure
loglog(nnew,I1berror,'ro',nnew,I2berror,'bo')
legend('For Function 1','For Function 2','location','northwest')
title('Plot Magnitude of Error versus 1/n')
ylabel('error')
xlabel('n')

% For the I(f1) verify that the error is decreasing like O(1/n^4).
x = linspace(4,128,6);
y = 1./(x.^4);

figure
plot(n,I1berror,'bo',x,y,'r-')
title('Error of I(f1) over n')
xlabel('n')
ylabel('error')
legend('error','1/x^4')
axis([0 128 -0.05 0.5])

% It looks as if I(f1) decreases like O(1/n^4)(Matlab says the
% equation to model this behavior would be f(n)=49.57*n^(-3.377) using
 the
% curve fitting tool)

% Does this rate of decrease in the error appear to be true for I(f2)?
% It does not seem to decrease like I(f1).
% What rate does the error appear to decrease for this integral?
% So I used the Matlab toolbox to fit a line to our I(f2) errors and
 it said
% the equation best fit would be f(n)=10.35*n^(-1.288) so I(f2)
 decreases like O(1/n)
end
```

```
T =

                 I1berror      I2berror
                 __________    _________

    f1 n=4         0.45914       1.7416
    f1 n=8        0.044179      0.69336
```

```
f1 n=16       0.0041705      0.29546
f1 n=32      0.00043743       0.1359
f2 n=64       4.968e-05     0.065165
f2 n=128     5.9078e-06     0.031909

prob5 : Success!
```




Plot Magnitude of Error versus 1/n
For Function 1
For Function 2
error
n
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-3}
10^{-2}
10^{-1}
10^{0}


Error of I(f1) over n
error
$1/x^4$
error
n
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-0.05
0
20
40
60
80
100
120

Published with MATLAB® R2016a