Table of Contents


```
% INSTRUCTIONS:
% 1) save this file into the directory that holds your work
% 2) make sure it runs without error
% 3) publish this file to PDF.  There are two ways to do this:
%    a) You can either use the 'Publish' tab, being sure to edit the
%    publishing options so the output format is 'pdf', or
%    b) run the commands below in the command window, one at a
%    time: (without the comment symbol!)
%    options.format = 'pdf';
%    publish('runAllHW9',options);
%
% 4) There should then be a PDF file in the subdirectory 'html'
%    that you can upload to TurnItIn. Make sure it has everything you
 need

% prepare to run...
clear all  % clear out all variables
close all  % close any open figures
```

Problem 1

```
type('MyPolyfit.m')
```

```
function [yFit,coeff,condNum,rSq]=MyPolyfit(x,y,polyOrder,isPlotted)
%{
Creates a fit for polynomial functions using given x,y and desired
poly order values and outputs a fitted graph, coeff matrix, condition
number and the R^2 value for the fit

Alyssa Rose  HW9  04-03-2018
%}

%% checks if vectors meet requirements
if length(x)~=length(y) || length(x) < polyOrder+1 || length(y) <
 polyOrder+1
    yFit = NaN;
    coeff = NaN;
    condNum = NaN;
```

```
    rSq = NaN;
    return
end

%forces isPlotted to be 0 if nothing passed in
if nargin < 4
    isPlotted = 0;
end

%% sets up A and solves for coeff
%forces x,y to be columns
y = y(:);
x = x(:);
b = y;
%creates matrix
z = polyOrder + 1;
A = zeros(length(x), polyOrder+1);
A(:,(1:z)) = x.^(polyOrder:-1:0);
condNum = cond(A);
coeff = A\b;

%finds yFit
yFit = A*coeff;

%% calcs. R^2 and plots
yAvg = mean(y);
ssTot = sum((y - yAvg).^2);
ssRes = sum((y - yFit).^2);
rSq = 1-(ssRes./ssTot);

if isPlotted ==1
    plot(x,yFit,'b',x,y,'ro')
    title(sprintf('Poly order %i , R^2 = %1.4f', polyOrder,rSq))
    xlabel('Independent Variable')
    ylabel('Dependent Variable')
end
end
```

Problem 2

```
type('TestPolyFit.m')

fprintf('\n----------------------------------------------\n')

TestPolyFit;


function [yTest,yFit]=TestPolyFit
%{
Tests MyPolyfit; outputs test vector and fitted vector

Alyssa Rose  HW9  04-03-2018
%}
```

```
%% creates x and yTest variables
x = [1:35];
yTest = (3*x.^4) + (1.5*x.^2) + 12;
polyOrder = 4;

%finds true, MyPolyfit and polyfit coeffs
P = [polyfit(x,yTest,polyOrder)]';
g = [3,0,1.5,0,12]';
[yFit,coeff,condNum,rSq]=MyPolyfit(x,yTest,polyOrder,0);
coeff = coeff';

%formats comparisons into table
fprintf('Power\tTrue\tMyPolyfit\tpolyfit\n')
for n = (0:polyOrder)
    fprintf('%i\t %8.2f%8.2f\t %8.2f\n',polyOrder - n, g(n+1) ,coeff(n
+1),P(n+1));
end
```

```
-------------------------------------------
Power True MyPolyfit polyfit
4      3.00    3.00      3.00
3      0.00   -0.00     -0.00
2      1.50    1.50      1.50
1      0.00   -0.00     -0.00
0     12.00   12.00     12.00
```

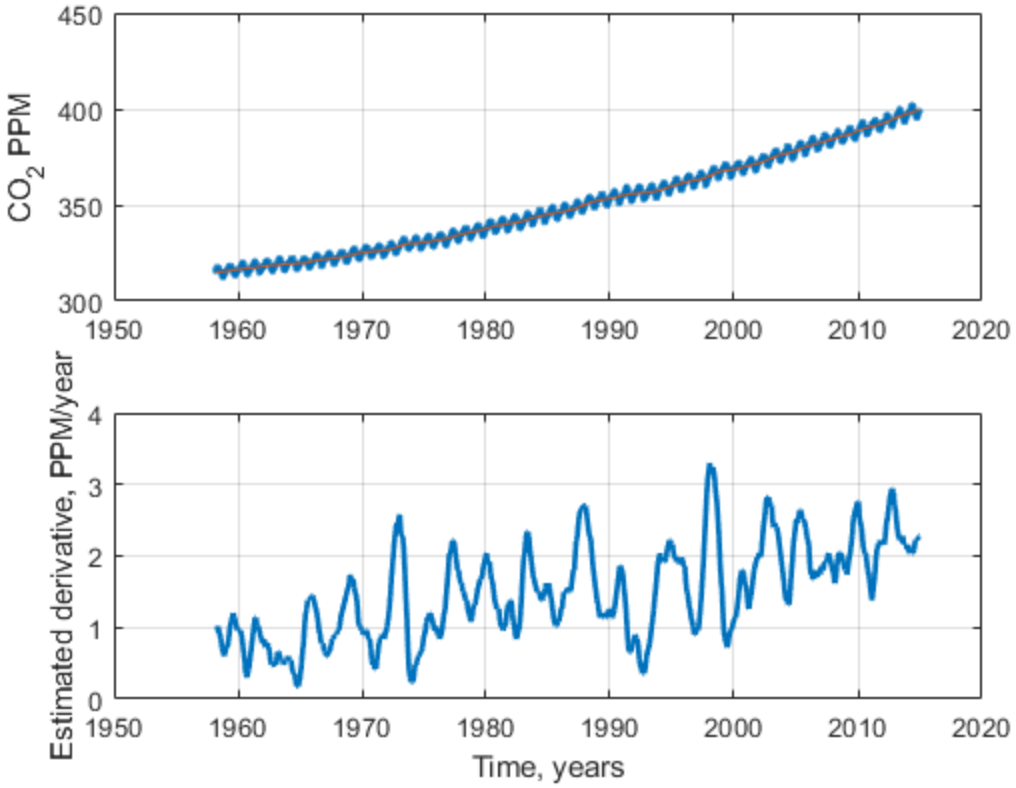
Problem 3

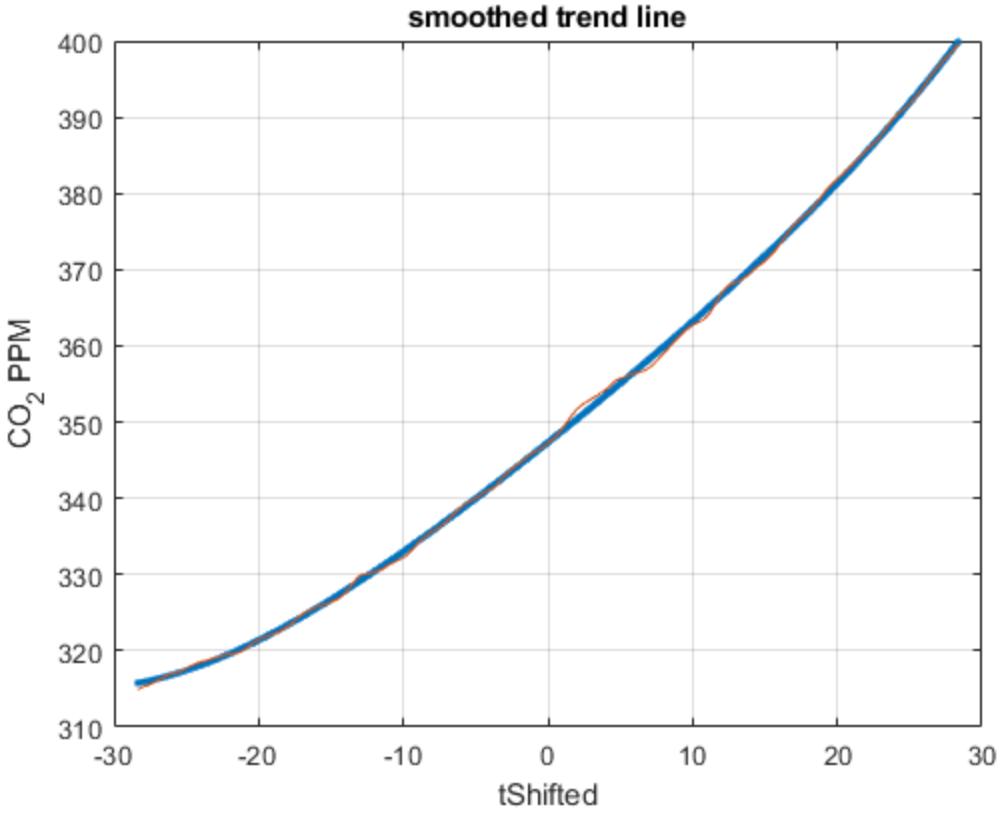
```
type('Co2Fits.m')
fprintf('\n-------------------------------------------\n')
Co2Fits


data = csvread('CO2_nohdr.csv');
%{
fits polynomial to CO2 data using centered and uncentered data

Alyssa Rose  HW9  04-05-2018
%}
%% creates unshifted/shifted fits for 3rd and 4th order polynomials
% pull out the data
t = data(:,4);
smoothedTrend = data(:,8);
rawData = data(:,9);
%computes non shifted fits
[raw3rdOrder,coeff3NC,condNum3NC,rSq3NC]=MyPolyfit(t,rawData,3,0);
[raw4thOrder,coeff2,condNum4NC,rSq4NC]=MyPolyfit(t,rawData,4,0);

%computes shifted fits
tShifted = t - mean(t);
[raw3rdShift,coeff3C,condNum3C,rSq3C]=MyPolyfit(tShifted,rawData,3,0);
[raw4thShift,coeff1,condNum4C,rSq4C]=MyPolyfit(tShifted,rawData,4,0);
```

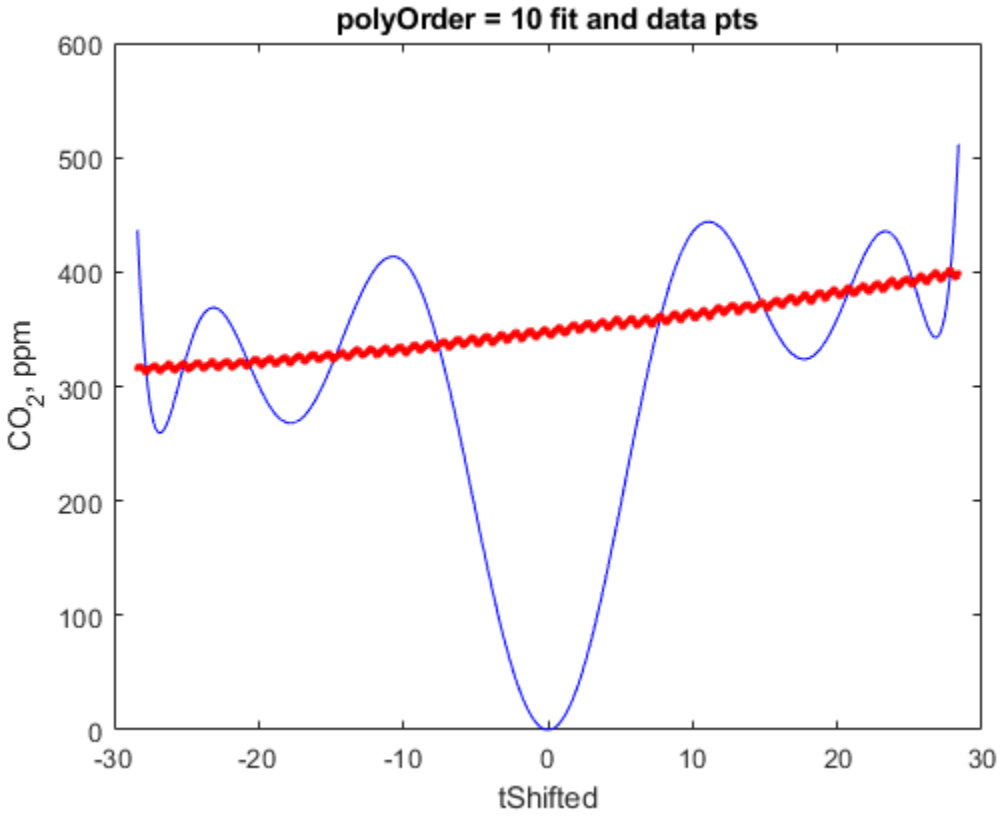
```
%% finds May 1, 2018 prediction
predTime = 2018+(5/12)-mean(t);
predic = (coeff1(1,1)*(predTime)^4) + (coeff1(2,1)*(predTime)^3) + ...
    (coeff1(3,1)*(predTime)^2) + (coeff1(4,1)*predTime) +
 (coeff1(5,1));
fprintf('prediction for May 1 2018: %.4f\n\n', predic);

%% plots data and smoothed trend line and the numerical derivative
%finds deriv w/ gradient
dt = mean(diff(t));
numericalDeriv = gradient(smoothedTrend)/dt;

%plots data
figure,
subplot(211)
plot(t,rawData,'.',t,smoothedTrend)
grid on
ylabel('CO_2 PPM')

subplot(212)
plot(t,numericalDeriv,'Linewidth',2)
grid on
xlabel('Time, years')
ylabel('Estimated derivative, PPM/year')

%% creates table of diff fits
coeff3C = [0;coeff3C];
coeff3NC = [0;coeff3NC];
fprintf('power\t\t4th Cen.\t\t4th\t\t3rd Cen.\t\t  3rd\n')
for p = (0:4)
    fprintf('%i\t\t\t\t%0.2f\t\t\t\t %0.2f\t\t   %0.2f\t\t     %0.2f
\n',4-p,coeff1(p+1),coeff2(p+1),coeff3C(p+1),coeff3NC(p+1))
end

%% plots of other fits
%plots smooth trend and shifted 4th order
figure(3)
plot(tShifted,raw4thShift,'.',tShifted,smoothedTrend)
grid on
ylabel('CO_2 PPM')
xlabel('tShifted')
title('smoothed trend line')

%plots polynomial fit of order 10
figure(4)
[fit10,coeff,condNum,rSq]=MyPolyfit(tShifted,rawData,10,0);
plot(tShifted,fit10,'b',tShifted,rawData,'r.')
title('polyOrder = 10 fit and data pts')
xlabel('tShifted')
ylabel('CO_2, ppm')

%plots centered/non centered and numerical deriv
figure(5)
```

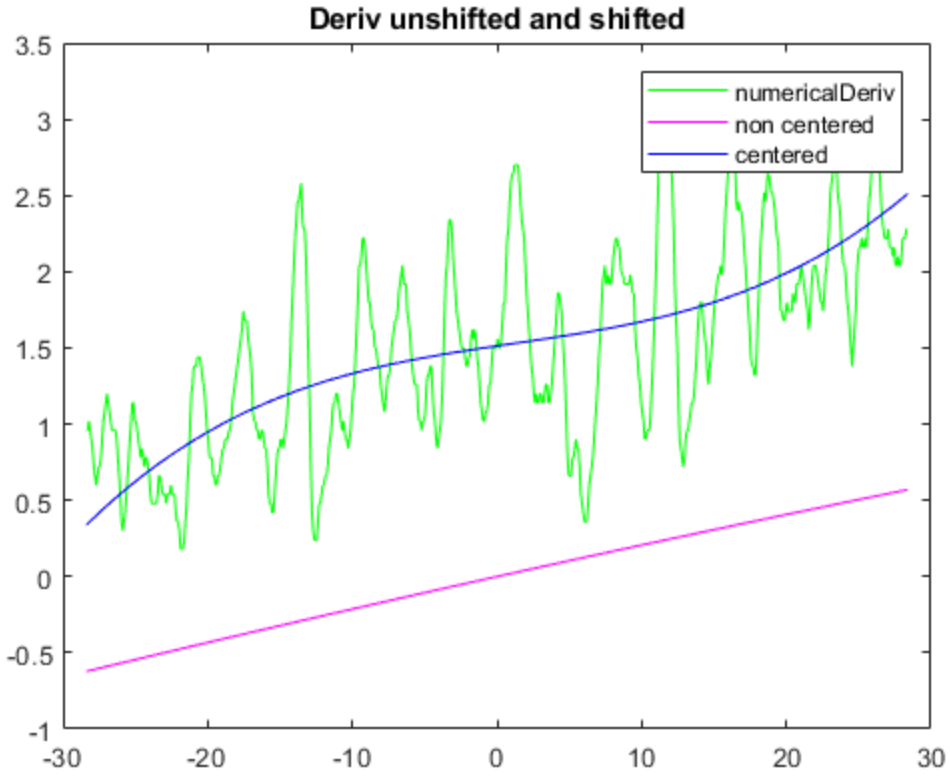
```
cenDeriv = (4*coeff1(1,1).*(tShifted).^3) +
 (3*coeff1(2,1).*(tShifted).^2) + ...
    (2*coeff1(3,1).*(tShifted)) + (coeff1(4,1));
nonCenDeriv = (4*coeff2(1,1).*(tShifted).^3) +
 (3*coeff2(2,1).*(tShifted).^2) + ...
    (2*coeff2(3,1).*(tShifted)) + (coeff2(4,1));
plot(tShifted,numericalDeriv,'g',tShifted,
 nonCenDeriv,'m',tShifted,cenDeriv,'b')
title('Deriv unshifted and shifted')
legend('numericalDeriv','non centered','centered');

%% creates table of cond. number and R^2 values
fprintf('PolyOrder\t Centered?\t R^2\t\t\t Condition#\n');
fprintf('3\t\t\t        No\t\t\t    %.4f\t\t\t%g\n',rSq3NC,condNum3NC)
fprintf('3\t\t\t        Yes\t\t    %.4f\t\t\t%g\n',rSq3C,condNum3C)
fprintf('4\t\t\t        No\t\t\t    %.4f\t\t\t%g\n',rSq4NC,condNum4NC)
fprintf('4\t\t\t        Yes\t\t    %.4f\t\t\t%g\n',rSq4C,condNum4C)

%% plots poly+sine fit [extra credit]
[yFit,coeff,RSq]=MyPolyPlusSinefit(t,rawData,4,1);
ylabel('CO_2, ppm')
xlabel('Time, years')
title(sprintf('RSq = %.6f', RSq))
legend('poly+sine fit','data')

%% discussion
%{
Centering the data creates a fit that has a higher R^2 value
and a smaller condition number; this data therefore is more accurate
and most closely fits the data.

Using R^2 and condition number for goodness-of-fit metrics
could be used to show that the 10th order poly fit is not accurate.

The centered and non centered derivatives differ since their
 differences
are being raised to high orders, which amplifies the differences.
Based on this data, it can be seen that CO_2 values are increasing
faster than a linear model (more cubic of a graph).
%}

-------------------------------------------
Warning: Rank deficient, rank = 3, tol =  3.102113e-02.
Warning: Rank deficient, rank = 3, tol =  6.165328e+01.
prediction for May 1 2018: 409.1504

power  4th Cen.  4th  3rd Cen.    3rd
4    0.00     0.00     0.00       0.00
3    -0.00     -0.00     -0.00       0.00
2    0.01     0.01     0.01       -0.02
1    1.51     0.00     1.51       21.82
0    347.33     0.00     346.91       0.00
Warning: Rank deficient, rank = 9, tol =  2.963017e+02.
PolyOrder  Centered?  R^2    Condition#
```

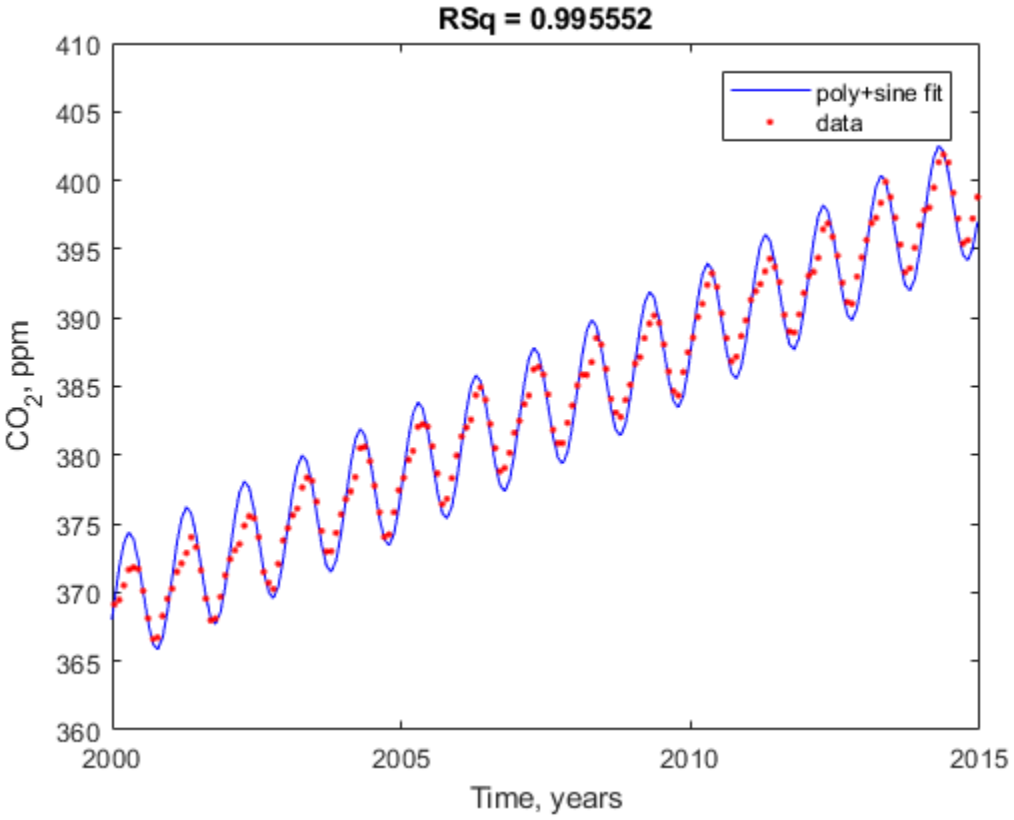
```
3           No       0.9922   1.77251e+16
3           Yes      0.9922   13009.6
4           No       0.9922   4.88663e+21
4           Yes      0.9924   407549
Warning: Rank deficient, rank = 3, tol =  6.165328e+01.
```




smoothed trend line
400
390
380
370
360
350
340
330
320
310
CO2 PPM
-30
-20
-10
0
10
20
30
tShifted


polyOrder = 10 fit and data pts
600
500
400
300
200
100
0
CO2, ppm
-30
-20
-10
0
10
20
30
tShifted

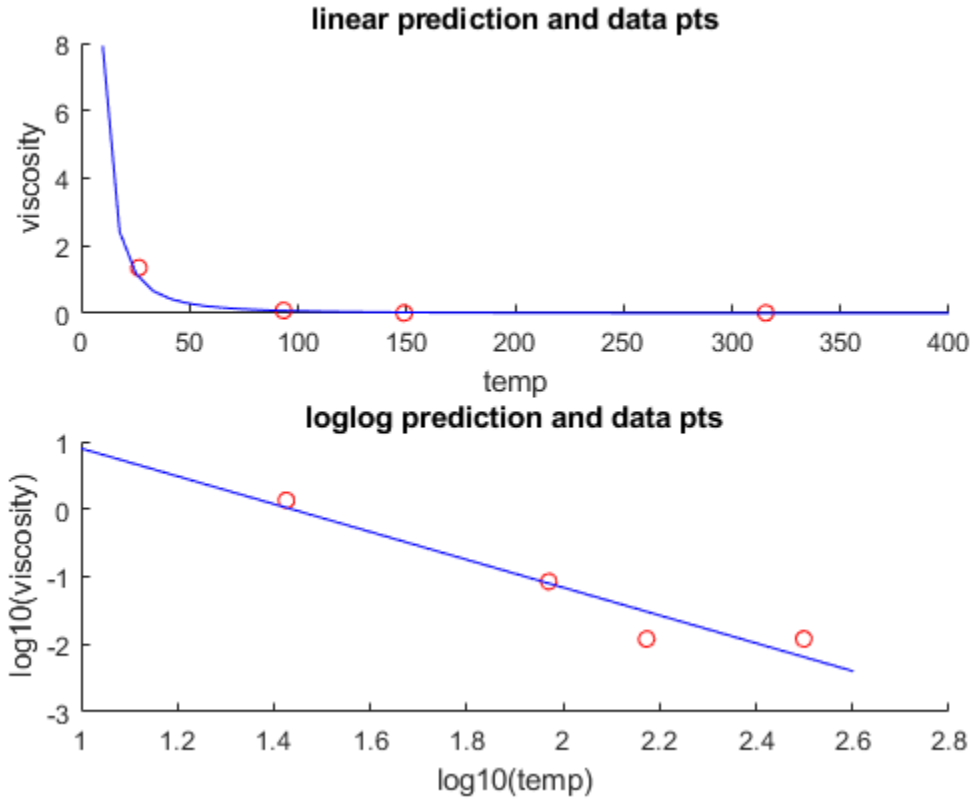

Deriv unshifted and shifted
numericalDeriv
non centered
centered
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-30
-20
-10
0
10
20
30


RSq = 0.995552
poly+sine fit
data
CO2, ppm
410
405
400
395
390
385
380
375
370
365
360
2000
2005
2010
2015
Time, years

Problem 4

```
type('ViscosityCurve');
fprintf('\n-----------------------------------------\n')
ViscosityCurve


function [alpha,beta]=ViscosityCurve
%{
fits linear to power curve and extrapolates; returns
slope and y intercept of line after transformation.

Alyssa Rose  HW9  04-05-2018
%}
%% creates data vecs
temp = [26.67, 93.33, 148.89, 315.56];
viscosity = [1.35, 0.085, 0.012, 0.012];
temp51 = linspace(10,400,51);

%% plots data pts and prediction
%"linear" plot
figure(3)
subplot(2,1,1)
hold on
[yFit, coeff] = MyPolyfit(log10(temp),log10(viscosity),1,0);
plot(temp,viscosity,'ro');

%finds beta and alpha values; uses it to make prediction
beta = coeff(1,1);
alpha = 10^(coeff(2,1));
predic = alpha*(temp51.^beta);
%plotting prediction
plot(temp51, predic,'b')
title('linear prediction and data pts')
xlabel('temp')
ylabel('viscosity')
hold off

%% plots log-log and polyfit
%log-log
subplot(2,1,2)
hold on
plot(log10(temp),log10(viscosity),'ro')
plot(log10(temp51),log10(predic),'b');
title('loglog prediction and data pts')
xlabel('log10(temp)')
ylabel('log10(viscosity)')
hold off

-----------------------------------------

ans =
```

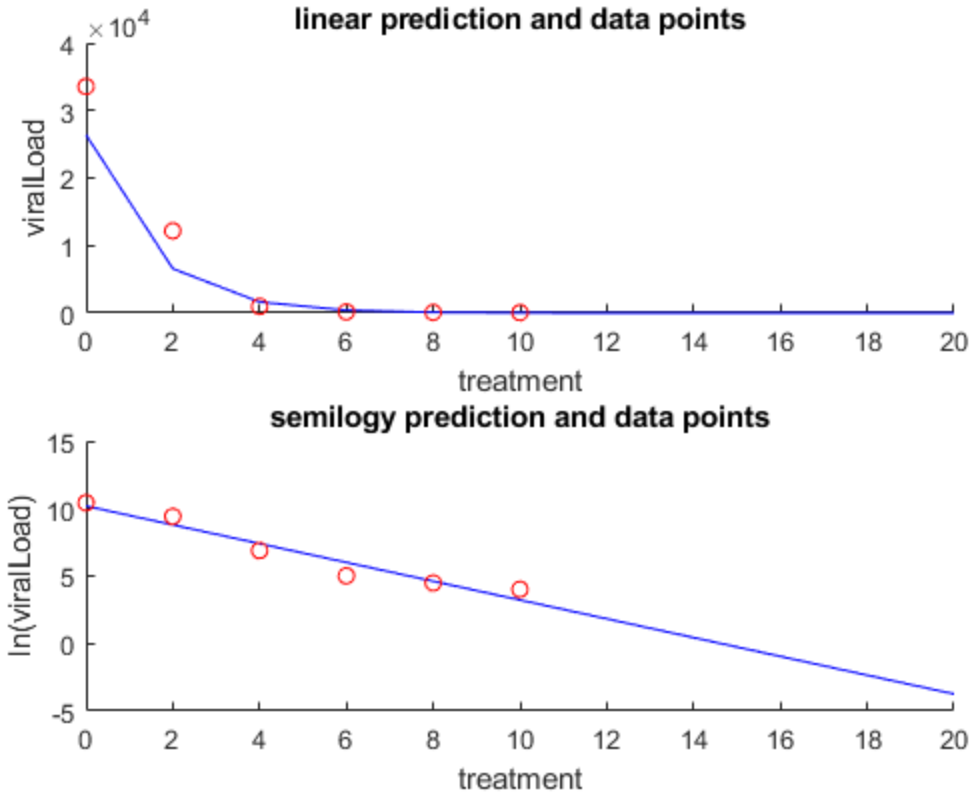
```
    906.3073
```



Problem 5

```
type('ViralLoad.m');
fprintf('\n----------------------------------------\n')
ViralLoad


function [alpha,beta]=ViralLoad
%{
fits linear to exponential curve and extrapolates; returns
slope and y intercept of line

Alyssa Rose  HW9  04-05-2018
%}

%% create vec, plot, find prediction
%creates data
treatment = 0:2:10;
viralLoad = [33550, 12170, 975, 150, 88, 55];

%"linear" plot
figure(4)
subplot(2,1,1)
```

```
hold on
[yFit, coeff] = MyPolyfit(treatment, log(viralLoad),1,0);

%finds alpha/beta and prediction and plots
beta = coeff(1,1);
alpha = exp(coeff(2,1));
treatPredic = 0:2:20;
linePredic = alpha*exp(beta.*treatPredic);
%plots prediction and data pts
plot(treatPredic, linePredic,'b');
plot(treatment, viralLoad,'ro');
title('linear prediction and data points')
ylabel('viralLoad')
xlabel('treatment')
hold off

%semilogy plot
subplot(2,1,2)
hold on
plot(treatPredic, log(linePredic), 'b')
plot(treatment, log(viralLoad),'ro')
title('semilogy prediction and data points')
xlabel('treatment')
ylabel('ln(viralLoad)')
hold off
%{
only semilogy can be used since treatment data contains a 0
in its vector, and ln(0) does not exist; this problem doesn't
occur in ViscosityCurve since log10(0) isn't a data point
%}


-------------------------------------------

ans =

   2.6407e+04
```



Extra credit - problem 6

```
try
    type('MyPolyPlusSinefit.m')
    % file run is at end of co2fit.m
catch
    disp('either no problem 6 file, or error running prob 6 code')
end


function [yFit,coeff,RSq]=MyPolyPlusSinefit(x,y,polyOrder,isPlotted)
%{
Does same thing as MyPolyFit but also includes adjustments for
sine and cosine
%}
%% checking input parameters
if length(x)~=length(y) || length(x) < polyOrder+1 || length(y) <
 polyOrder+1
    yFit = NaN;
    coeff = NaN;
    condNum = NaN;
    RSq = NaN;
    return
end

if nargin < 4
```

```
    isPlotted = 0;
end
%forces data to columns
y = y(:);
x = x(:);
b = y;

%% creates matrix A and additional terms
z = polyOrder + 3;
A(:,(1:z-2)) = x.^(polyOrder:-1:0);
condNum = cond(A);
coeff = A\b;
%creates sin and cosine adjustments
A(:,z-1) = sin(2*pi*x);
A(:,z) = cos(2*pi*x);
H=[A(:,z-1),A(:,z)];
K = H\b;
a = A(:,(1:z-2))*coeff;
s = A(:,z-1:z)*K;
yFit = [a+s];

%% computinf R^2 and plotting
yAvg = mean(y);
ssTot = sum((y - yAvg).^2);
ssRes = sum((y - yFit).^2);
RSq = 1-(ssRes./ssTot);

if isPlotted ==1
    figure(6)
    plot(x,yFit,'b',x,y,'r.')
    axis([2000 2015 360 410]);
end
end
```

Published with MATLAB® R2017b