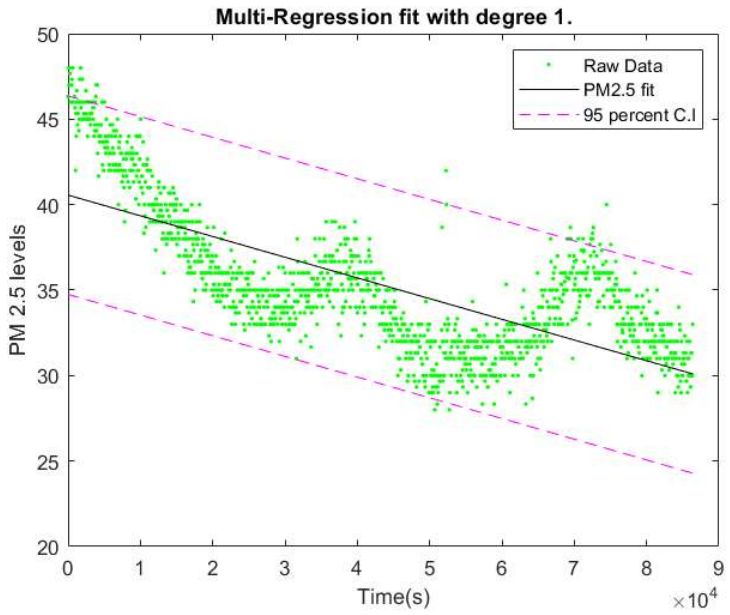
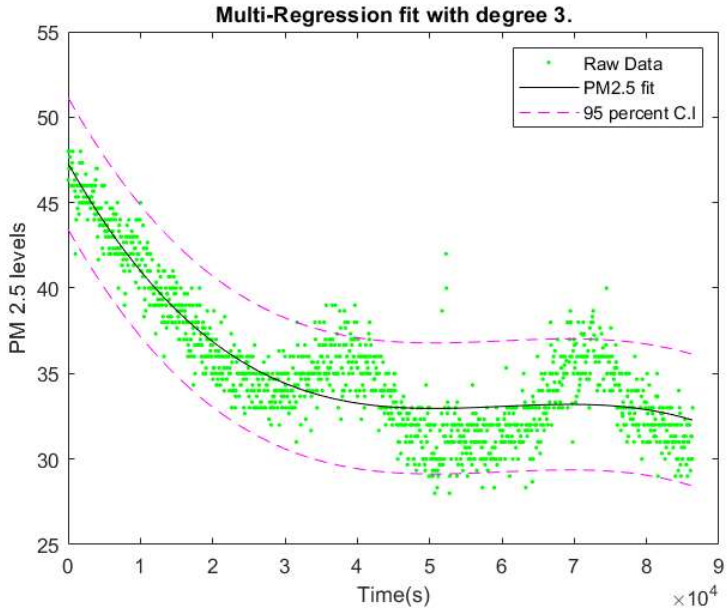
Part I [30 points]: Linear Regression.

```
%Part 1
clear
load hw3_1.mat
pm2d5= data.pm2d5;
time = data.time;
time_second = (datenum(time)-floor(datenum(time)))*24*60*60;

figure
degree= 1;
[p1, SI]= polyfit(normalize(time_second,'zscore'), pm2d5,degree);
[yfit, delta] = polyval(p1, normalize(time_second,'zscore'), SI);
plot(time_second, pm2d5, '.g');
hold on
plot(time_second, yfit, 'k');
plot(time_second,yfit+2*delta,'m--',time_second,yfit-2*delta,'m--');
header = sprintf("Multi-Regression fit with degree %.0f.", degree);
title(header), xlabel('Time(s)'), ylabel('PM 2.5 levels'), legend('Raw Data','PM2.5 fit','95 percent C.I');
hold off
```



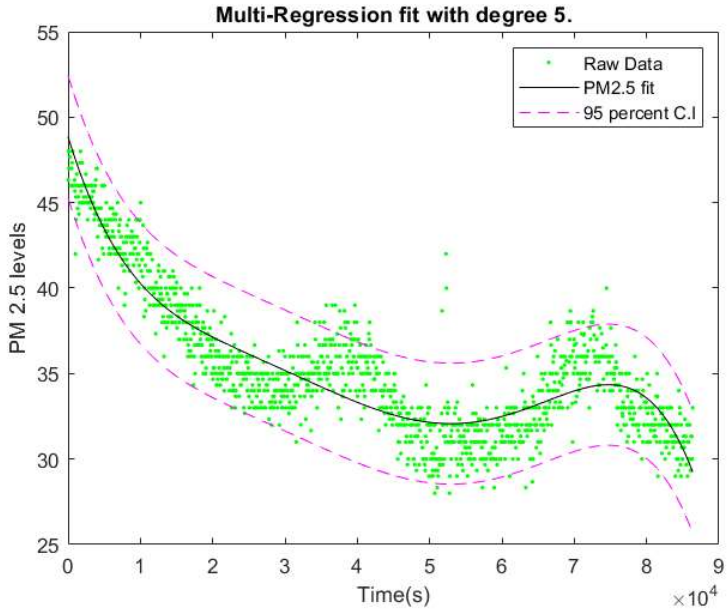
a) See above.

```
figure
degree= 3;
[p1, SI]= polyfit(normalize(time_second,'zscore'), pm2d5,degree);
[yfit, delta] = polyval(p1,normalize(time_second,'zscore'), SI);
plot(time_second, pm2d5, '.g');
hold on
plot(time_second, yfit, 'k');
plot(time_second,yfit+2*delta,'m--',time_second,yfit-2*delta,'m--');
header = sprintf("Multi-Regression fit with degree %.0f.", degree);
title(header), xlabel('Time(s)'), ylabel('PM 2.5 levels'), legend('Raw Data','PM2.5 fit','95 percent C.I');
hold off
```



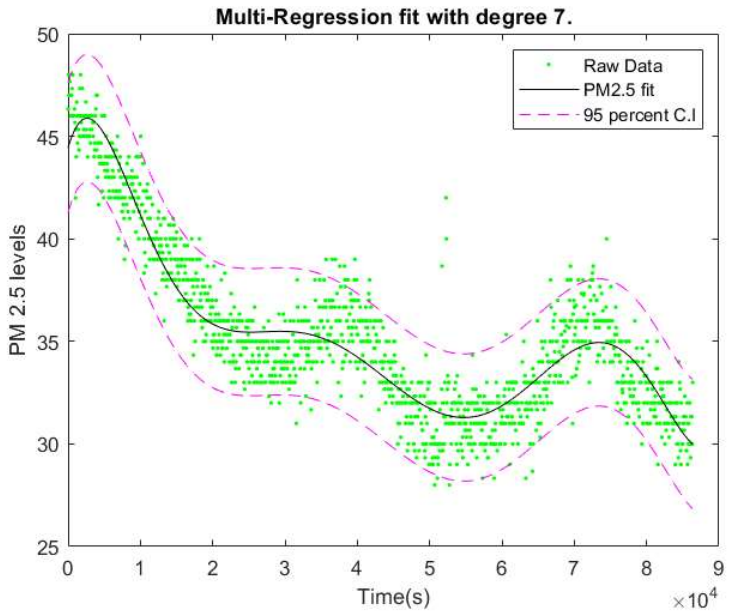
b) Polynomial degree 3 seems to capture the overall trend of PM 2.5 data (changes in local gradients) much better than the linear fit (degree 1).

```
figure
degree= 5;
[p1, SI]= polyfit(normalize(time_second,'zscore'), pm2d5,degree);
[yfit, delta] = polyval(p1, normalize(time_second,'zscore'), SI);
plot(time_second, pm2d5, '.g');
hold on
plot(time_second, yfit, 'k');
plot(time_second,yfit+2*delta,'m--',time_second,yfit-2*delta,'m--');
header = sprintf("Multi-Regression fit with degree %.0f.", degree);
title(header), xlabel('Time(s)'), ylabel('PM 2.5 levels'), legend('Raw Data','PM2.5 fit','95 percent C.I');
hold off
```




```
figure
degree= 7;
[p1, SI]= polyfit(normalize(time_second,'zscore'), pm2d5,degree);
[yfit, delta] = polyval(p1, normalize(time_second,'zscore'), SI);
plot(time_second, pm2d5, '.g');
hold on
plot(time_second, yfit, 'k');
plot(time_second,yfit+2*delta,'m--',time_second,yfit-2*delta,'m--');
header = sprintf("Multi-Regression fit with degree %.0f.", degree);
```

```
title(header), xlabel('Time(s)'), ylabel('PM 2.5 levels'), legend('Raw Data','PM2.5 fit','95 percent C.I');
hold off
```



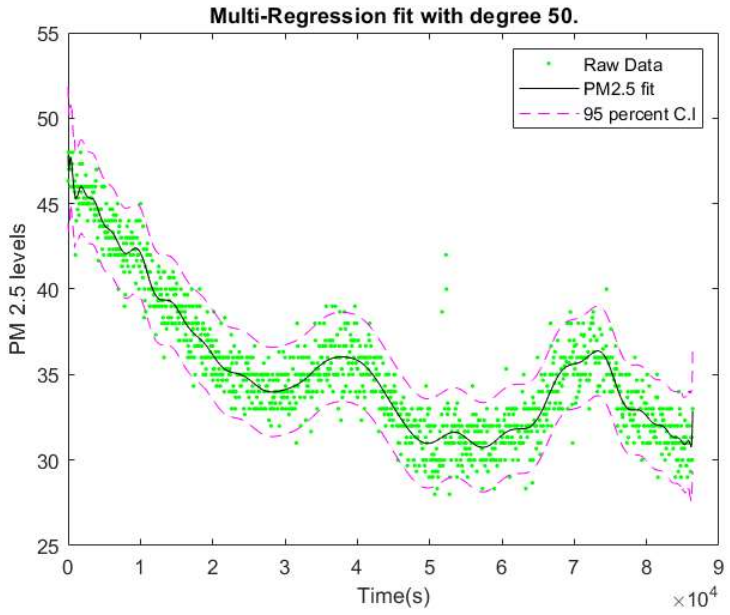

```
figure
degree= 50;
[p1, SI]= polyfit(normalize(time_second,'zscore'), pm2d5,degree);
```

```
Warning: Polynomial is badly conditioned. Add points with distinct X values, reduce the degree of the polynomial, or
try centering and scaling as described in HELP POLYFIT.
```

```
[yfit, delta] = polyval(p1, normalize(time_second,'zscore'), SI);
```

```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =  1.331952e-17.
```

```
plot(time_second, pm2d5, '.g');
hold on
plot(time_second, yfit, 'k');
plot(time_second,yfit+2*delta,'m--',time_second,yfit-2*delta,'m--');
header = sprintf("Multi-Regression fit with degree %.0f.", degree);
title(header), xlabel('Time(s)'), ylabel('PM 2.5 levels'), legend('Raw Data','PM2.5 fit','95 percent C.I');
hold off
```

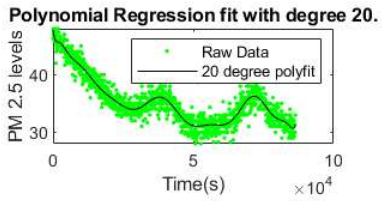


c) As the polynomial degree increases from 1 to 7, the fit seems to capture the trend and seasonality of data much better. However, when you increase the degree to 50, the polynomial seems to capture the potential noise spikes too, which is not desirable.

d) I believe degree 7 polynomial fit seems to represent the data the best because it captures the overall trend and seasonality without overfitting to the data noises. The benefit of the higher degree polynomial fit is that it captures the micro changes in seasonality and trend of the observation data. However, after a certain point, the high degree polynomial seems to overfit the potential noises in the data, at which point the fit becomes non-generalizable to unseen data.

Part II [30 points]: Regularization + Other Factors

```
%Part 2

%20 degree polynomial fit
figure
subplot(3,2,1)
degree= 20;
[p1, SI]= polyfit(normalize(time_second,'zscore'), pm2d5,degree);
[yfit_20, delta] = polyval(p1, normalize(time_second,'zscore'), SI);
plot(time_second, pm2d5, '.g');
hold on
plot(time_second, yfit_20, 'k');
header = sprintf("Polynomial Regression fit with degree %.0f.", degree);
title(header), xlabel('Time(s)'), ylabel('PM 2.5 levels'), legend('Raw Data','20 degree polyfit');
hold off
```



a) See above.

b) The cost function of Ridge Regression is the Residual Sum Squares + Regularization factor:

$$\hat{\beta} = \text{argmin}\left\{\sum_{i=1}^{N}\left(y_i - \sum_{j=0}^{n} x_{ij}\beta_j\right)^2 + \lambda\sum_{j=1}^{n}{\beta_j}^2\right\}$$

where N = number of data samples, n = number of input features, β = learnable coefficients,
$\hat{\beta}$ = optimal coefficient that minimizes cost function, x = input data, and y = ground truth output data
λ = Regularization term

The equation has a closed form solution for polynomial curve fitting:

$$\hat{\beta} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$$

where:

$$\mathrm{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{20} \\ 1 & x_2 & x_2^2 & \dots & x_2^{20} \\ 1 & x_3 & x_3^2 & \dots & x_3^{20} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^{20} \end{bmatrix}$$ and Y= Ground Truth PM2.5 data from y_1 to y_n

c) The cost function of LASSO regression:

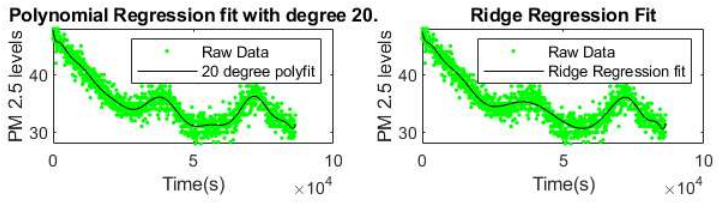
$$\hat{\beta} = \text{argmin}\left\{\sum_{i=1}^{N}\left(y_i - \sum_{j=0}^{n} x_{ij}\beta_j\right)^2 + \lambda\sum_{j=1}^{n}|\beta_j|\right\}$$

where N = number of data samples, n = number of input features, β = learnable coefficients,
$\hat{\beta}$ = optimal coefficient that minimizes cost function, x = input data, and y = ground truth output data
λ = Regularization term

From the cost function equation we can see that LASSO is different from Ridge Regression because the Regularization term is multiplied to the absolute value of the sum of coefficients rather than the square of the sum of coefficients. This helps with dimensionality reduction of input features and also reduces the effects of outlier data on optimal coefficient determination.

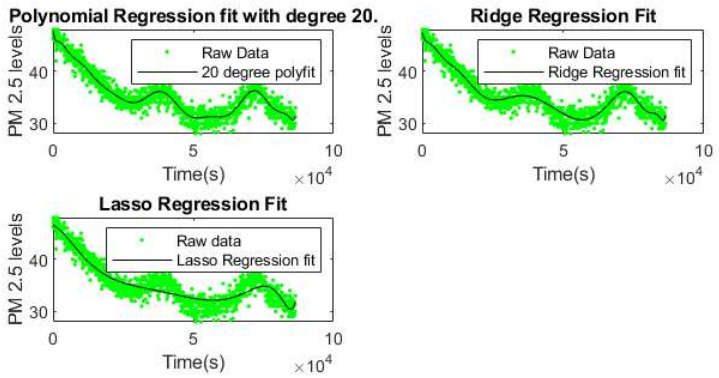
This cost function does NOT have a closed form solution like Ridge Regression.

```
%Creating Vandermonde Matrix of degree 20
dim= 20;
num_d= length(time_second);
V_m= ones(num_d,dim+1);
for i= 1:dim+1
    V_m(:,i)= normalize(time_second,'zscore').^(dim+1-i);
end

%Ridge Regression
Y= pm2d5;
lambda = 0.1;
dim = size(V_m,2);
beta = inv((V_m'*V_m)+lambda*eye(dim))*V_m'*Y;
y_fit_ridge = V_m*beta;
subplot(3,2,2)
plot(time_second,Y,'.g')
hold on
plot(time_second, y_fit_ridge,'k')
title('Ridge Regression Fit'), xlabel('Time(s)'), ylabel('PM 2.5 levels'), legend('Raw Data','Ridge Regression fit');
hold off
```



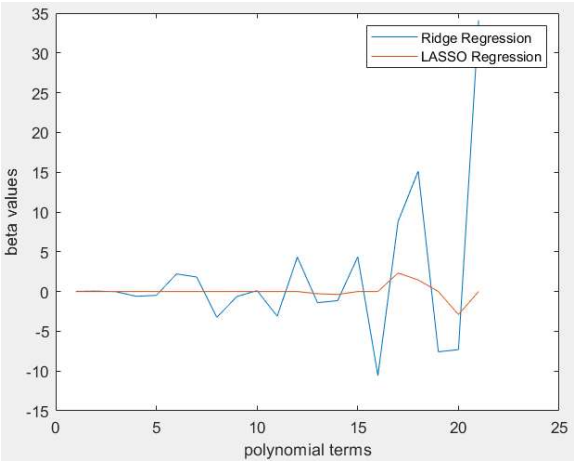

```
%LASSO
lambda=0.01;
[Beta, Fitinfo] = lasso(V_m,Y,'Lambda',lambda);
intercept=Fitinfo.Intercept;
y_fit_lasso = V_m*Beta+intercept;
subplot(3,2,3)
plot(time_second,Y,'.g')
hold on
plot(time_second, y_fit_lasso,'k')
title('Lasso Regression Fit'), xlabel('Time(s)'), ylabel('PM 2.5 levels'), legend('Raw data','Lasso Regression fit');
```

```
hold off
```



d) $\lambda = 0.1$ seems to work best for the Ridge Regression fit and $\lambda = 0.01$ works best for the LASSO fit. The Ridge Regression and LASSO fits seem to work better with the overall trend, without overfitting to the outliers data- especially around $5 \times 10^4 s$.

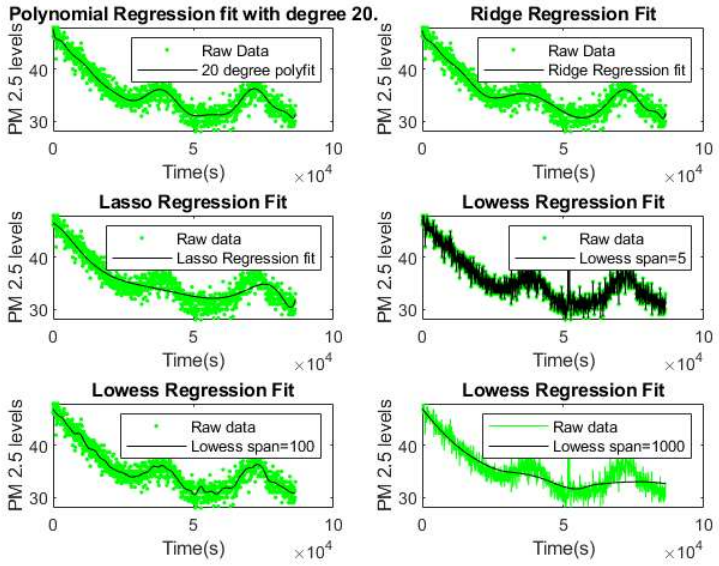
LASSO regression's coefficients seem much more constant around 0 compared to Ridge Regression's coefficients:



This is likely the reason why LASSO fit's plot looks underfitted compared to Ridge Regression's fit because LASSO has more 0 values- hence more dimensions reduced.

```
%LOWESS
span = 5;
ysmooth =smooth(time_second,Y,span,'loess');
subplot(3,2,4)
plot(time_second, Y,'.g')
hold on
plot(time_second, ysmooth,'k')
title('Lowess Regression Fit'), xlabel('Time(s)'), ylabel('PM 2.5 levels'), legend('Raw data','Lowess span=5');
hold off

span = 100;
ysmooth =smooth(time_second,Y,span,'loess');
subplot(3,2,5)
plot(time_second, Y,'.g')
hold on
plot(time_second, ysmooth,'k')
title('Lowess Regression Fit'), xlabel('Time(s)'), ylabel('PM 2.5 levels'), legend('Raw data','Lowess span=100');
hold off

span = 1000;
ysmooth =smooth(time_second,Y,span,'loess');
```

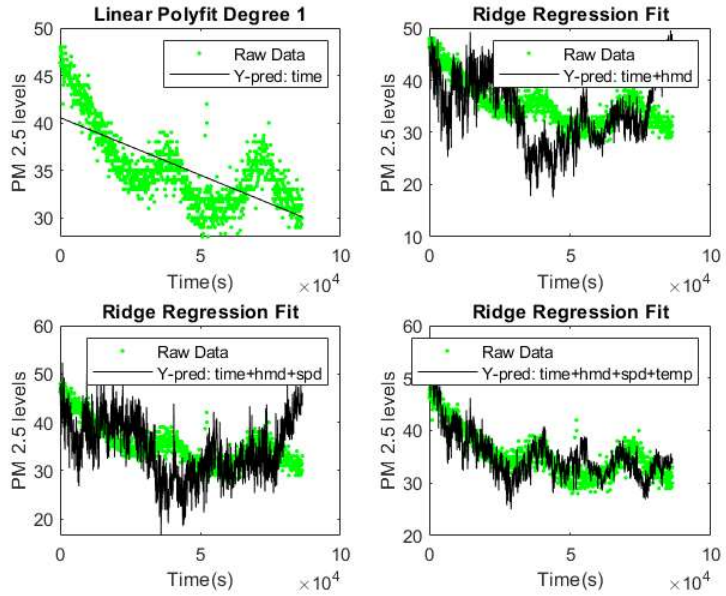
```
subplot(3,2,6)
plot(time_second, Y,'g')
hold on
plot(time_second, ysmooth,'k')
title('Lowess Regression Fit'), xlabel('Time(s)'), ylabel('PM 2.5 levels'), legend('Raw data','Lowess span=1000');
hold off
```



e) From the last 3 subplots, we see that as the span of LOWESS smoothing increases its fitting to raw data decreases. LOWESS with span 5 seems to excessively overfit the data while span 1000 seems to miss some changes in overall data trend. Hence, LOWESS smoothing with span=100 seems to best represent the data which captures the overall trend without overfitting.

```
%Linear Regression with multiple features using Ridge Regression

lambda =0.1;
figure
subplot(2,2,1)
degree= 1;
[p1, SI]= polyfit(normalize(time_second,'zscore'), pm2d5,degree);
[yfit, delta] = polyval(p1, normalize(time_second,'zscore'), SI);
plot(time_second, pm2d5, '.g');
hold on
plot(time_second, yfit, 'k');
title('Linear Polyfit Degree 1'), xlabel('Time(s)'), ylabel('PM 2.5 levels'), legend('Raw Data','Y-pred: time');
hold off

X= [normalize(time_second,'range'), normalize(data.hmd,'range')];
Y= pm2d5;
dim = size(X,2);
beta = inv((X'*X)+lambda*eye(dim))*X'*Y;
y_fit_ridge = X*beta;
subplot(2,2,2)
plot(time_second,Y,'.g')
hold on
plot(time_second, y_fit_ridge,'k')
title('Ridge Regression Fit'), xlabel('Time(s)'), ylabel('PM 2.5 levels'), legend('Raw Data','Y-pred: time+hmd');
hold off

X= [normalize(time_second,'range'), normalize(data.hmd,'range'),normalize(data.spd,'range')];
dim = size(X,2);
beta = inv((X'*X)+lambda*eye(dim))*X'*Y;
y_fit_ridge = X*beta;
subplot(2,2,3)
plot(time_second,Y,'.g')
hold on
plot(time_second, y_fit_ridge,'k')
title('Ridge Regression Fit'), xlabel('Time(s)'), ylabel('PM 2.5 levels'), legend('Raw Data','Y-pred: time+hmd+spd');
hold off
```

```
X= [normalize(time_second,'range'), normalize(data.hmd,'range'),normalize(data.spd,'range'),normalize(data.tmp,'range')];
dim = size(X,2);
beta = inv((X'*X)+lambda*eye(dim))*X'*Y;
y_fit_ridge = X*beta;
subplot(2,2,4)
plot(time_second,Y,'.g')
hold on
plot(time_second, y_fit_ridge,'k')
title('Ridge Regression Fit'), xlabel('Time(s)'), ylabel('PM 2.5 levels'), legend('Raw Data','Y-pred: time+hmd+spd+temp');
hold off
```



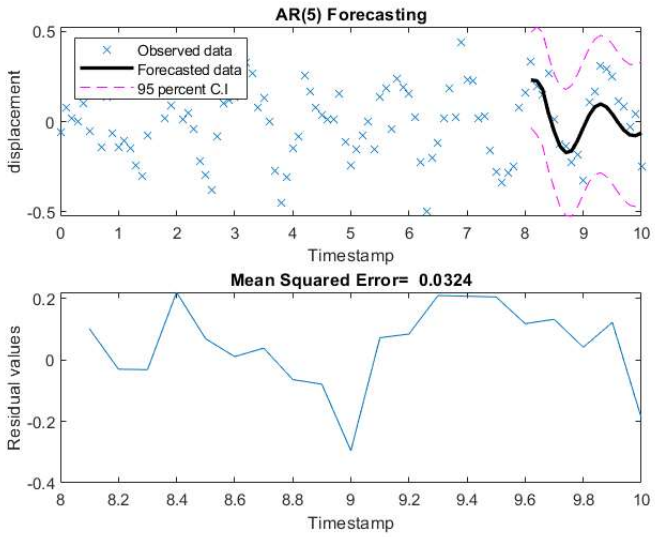
f) From the plots above, we see that as we add more features to the data, the linear fit seems to improve. While adding humidity and wind speed features to time seems to improve the fit compared to just time, adding temperature to the features seems to best improve the fit of the data.The last 2 plots prove this statement.

Part III [30 points]: Autoregressive model

```
clear
load hw3_3.mat

y= displacement(:,2);
time= displacement(:,1);

% AR(5) Model:
order=5;
Mdl= arima(order,0,0);
EstMdl = estimate(Mdl,y(1:81));
```

```
ARIMA(5,0,0) Model (Gaussian Distribution):

               Value       StandardError    TStatistic      PValue
              _________    _____________    __________    _________

Constant     -0.0084379       0.016523       -0.51069       0.60957
AR{1}          0.52783         0.11849         4.4548     8.3976e-06
AR{2}         0.097503         0.13631        0.71532        0.47441
AR{3}        -0.095204         0.17144       -0.55532        0.57868
AR{4}         -0.10412         0.14563       -0.71496        0.47463
AR{5}         -0.27807         0.12547        -2.2162       0.026676
Variance      0.017584       0.0035426         4.9636     6.9202e-07
```

```
[Y,YMSE] = forecast(EstMdl,20,y(1:81));
figure
subplot(2,1,1)
plot(time, y,'x');hold on;
plot([8.1:0.1:10],Y,'k','LineWidth',2);
plot([8.1:0.1:10],Y+2*YMSE.^0.5,'m--',[8.1:0.1:10],Y-2*YMSE.^0.5,'m--');
title('AR(5) Forecasting'), xlabel('Timestamp'), ylabel('displacement'), legend({'Observed data','Forecasted data','95 percent C.I'},'Loca
hold off
subplot(2,1,2)
Residual= y(82:101)-Y;
plot([8.1:0.1:10],Residual)
MSE = mean(YMSE); header= sprintf("Mean Squared Error=  %.4f", MSE);
title(header), xlabel('Timestamp'), ylabel('Residual values');
```



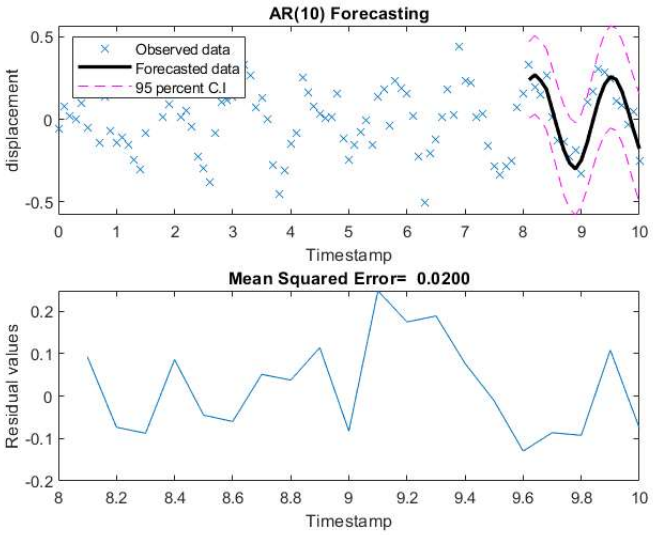
a) See above. **MSE= 0.0324**

```
% AR(10) Model:
clear
load hw3_3.mat

y= displacement(:,2);
time= displacement(:,1);
order=10;
Mdl= arima(order,0,0);
EstMdl = estimate(Mdl,y(1:81));
```

```
ARIMA(10,0,0) Model (Gaussian Distribution):
```

```
              Value      StandardError    TStatistic      PValue
              _______    _____________    __________    _________

Constant     -0.021082       0.016457        -1.281        0.20018
AR{1}          0.24929        0.12751        1.9551       0.050574
AR{2}       -0.045623        0.14478      -0.31512        0.75267
AR{3}        -0.22532        0.15358       -1.4671        0.14235
AR{4}        -0.17653        0.13418       -1.3156         0.1883
AR{5}        -0.16472        0.13804       -1.1933        0.23277
AR{6}        -0.28952        0.10689       -2.7087      0.0067543
AR{7}         -0.1765        0.11302       -1.5617        0.11835
AR{8}        -0.16147          0.124       -1.3022        0.19283
AR{9}        -0.17116        0.13362        -1.281        0.20021
```

```
[Y,YMSE] = forecast(EstMdl,20,y(1:81));
figure
subplot(2,1,1)
plot(time, y,'x');hold on;
plot([8.1:0.1:10],Y,'k','LineWidth',2);
plot([8.1:0.1:10],Y+2*YMSE.^0.5,'m--',[8.1:0.1:10],Y-2*YMSE.^0.5,'m--');
title('AR(10) Forecasting'), xlabel('Timestamp'), ylabel('displacement'), legend({'Observed data','Forecasted data','95 percent C.I'},'Loc
hold off
subplot(2,1,2)
Residual= y(82:101)-Y;
plot([8.1:0.1:10],Residual)
MSE = mean(YMSE); header= sprintf("Mean Squared Error=  %.4f", MSE);
title(header), xlabel('Timestamp'), ylabel('Residual values');
```

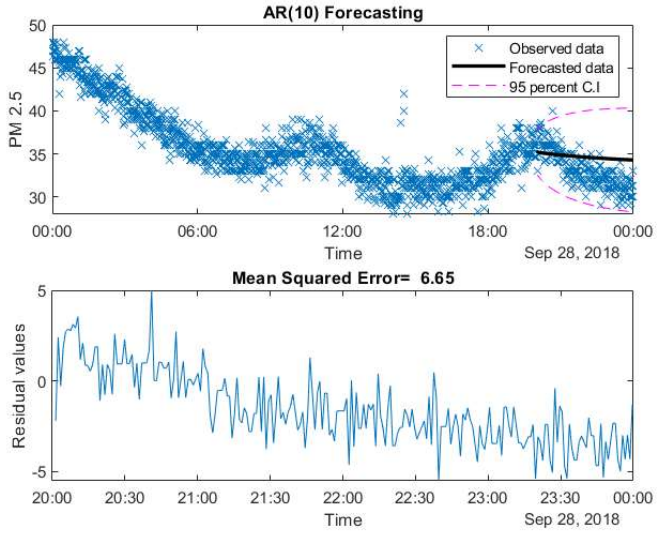


b) AR(10) forecasting seems to fit the observed data much better than AR(5) as can be inspected visually and corroborated by the lower **MSE=0.020**. AR(10) therefore provides a better prediction- also because of the fact that each forecasted value is estimated using the regression fit of last 10 values compared to AR(5) that uses last 5 values.

```
clear
load hw3_1.mat
pm2d5 = data.pm2d5;
time = data.time;
time_second = (datenum(time)-floor(datenum(time)))*24*60*60;

% AR(10) Model:
order=10;
Mdl= arima(order,0,0);
EstMdl = estimate(Mdl,pm2d5(1:1200));
```

```
ARIMA(10,0,0) Model (Gaussian Distribution):

              Value      StandardError    TStatistic      PValue
              _______    _____________    __________    _________

Constant      0.73578         0.35684         2.062       0.039211
AR{1}         0.19845        0.023467        8.4564      2.7564e-17
AR{2}         0.10403        0.028445        3.6571      0.00025507
AR{3}         0.10943        0.031437        3.4809      0.00049982
AR{4}         0.10402        0.031022        3.3531      0.00079916
AR{5}        0.040042        0.030008        1.3344         0.18207
AR{6}        0.094724         0.03027        3.1293       0.0017524
AR{7}        0.092292        0.028987        3.1839        0.001453
AR{8}        0.021356        0.025612       0.83385         0.40436
AR{9}         0.13271         0.02242        5.9191      3.2367e-09
```

```
[Y,YMSE] = forecast(EstMdl,238,pm2d5(1:1200));

figure
subplot(2,1,1)
plot(time, pm2d5,'x');hold on;
plot(time(1201:end),Y,'k','LineWidth',2);
plot(time(1201:end),Y+2*YMSE.^0.5,'m--',time(1201:end),Y-2*YMSE.^0.5,'m--');
title('AR(10) Forecasting'), xlabel('Time'), ylabel('PM 2.5'), legend({'Observed data','Forecasted data','95 percent C.I'},'Location','nor
hold off
subplot(2,1,2)
Residual= pm2d5(1201:end)-Y;
plot(time(1201:end),Residual)
MSE = mean(YMSE); header= sprintf("Mean Squared Error=  %.2f", MSE);
title(header), xlabel('Time'), ylabel('Residual values');
```

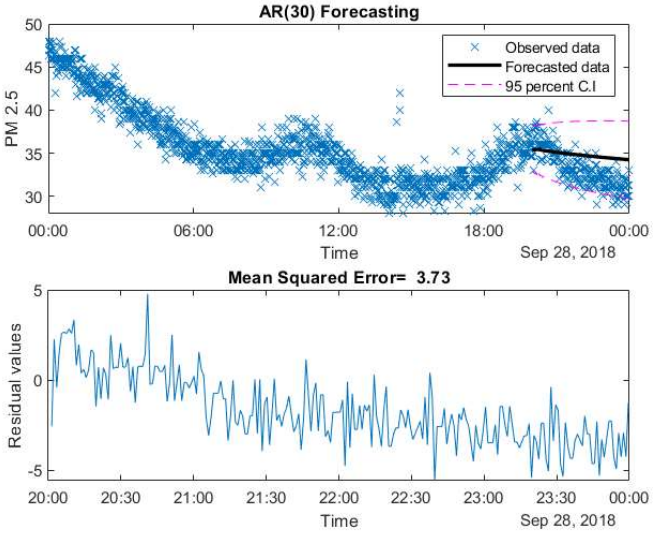



```
% AR(30) Model:
order=30;
Mdl= arima(order,0,0);
EstMdl = estimate(Mdl,pm2d5(1:1200));
```

ARIMA(30,0,0) Model (Gaussian Distribution):

	Value	StandardError	TStatistic	PValue
Constant	0.71739	0.33396	2.1481	0.031703
AR{1}	0.18226	0.025037	7.2797	3.3447e-13
AR{2}	0.082829	0.030968	2.6747	0.0074804
AR{3}	0.082057	0.032402	2.5324	0.011328
AR{4}	0.080243	0.033013	2.4307	0.015071
AR{5}	0.012776	0.033057	0.38647	0.69915
AR{6}	0.069641	0.031311	2.2241	0.02614
AR{7}	0.068386	0.030876	2.2148	0.026772
AR{8}	-0.014202	0.027984	-0.50749	0.61181

```
[Y,YMSE] = forecast(EstMdl,238,pm2d5(1:1200));

figure
subplot(2,1,1)
plot(time, pm2d5,'x');hold on;
plot(time(1201:end),Y,'k','LineWidth',2);
plot(time(1201:end),Y+2*YMSE.^0.5,'m--',time(1201:end),Y-2*YMSE.^0.5,'m--');
title('AR(30) Forecasting'), xlabel('Time'), ylabel('PM 2.5'), legend({'Observed data','Forecasted data','95 percent C.I'},'Location','nor
hold off
subplot(2,1,2)
Residual= pm2d5(1201:end)-Y;
plot(time(1201:end),Residual)
MSE = mean(YMSE); header= sprintf("Mean Squared Error=  %.2f", MSE);
title(header), xlabel('Time'), ylabel('Residual values');
```



c) As seen from the AR(10) and AR(30) plots above and their corresponding residual plots, the forecasted PM 2.5 values seem to severely underperform/underfit compared to harmonic displacement vaues in part b). This conclusion made from visual inspection of the plots is confirmed by the high MSE values of 6.65 for AR(10) and 3.73 for AR(30). I don't think increasing the order of AR model would help on improving the prediction performance because the result of increasing model order from 10 to 30 did very little to improve the prediction accuracy as can be seen visually and through MSE scores. There seems to be a diminishing return of MSE from increasing the model order. Therefore, I do not think that AR model is suitable for predicting PM 2.5 - most likley because of the fact that there are too many environmental features that affect the PM 2.5 levels and AR seems to work the best with low feature data.