

TEST 3 REGULARIZED LINEAR REGRESSION

Patel, Femina

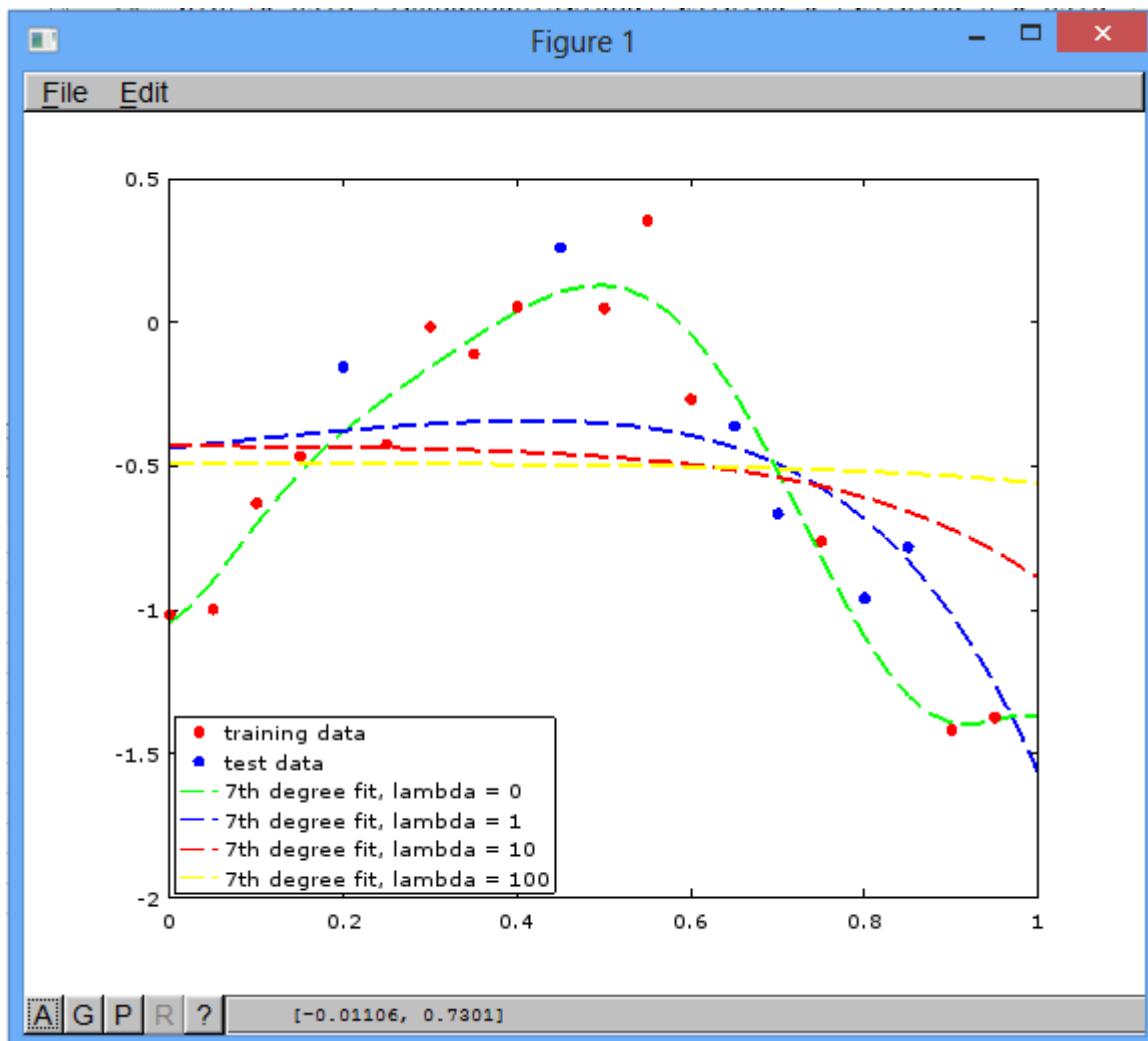
ID: N03094115

Answer 1:

```
function v = LinRegRegularized(xData,yData,x_vals,n,lambda)
m = length(xData);
m2 = length(x_vals);
for i=1:m
    for j=0:n
        X(i,j+1) = xData(i)^j;
    end
end
dg=eye(n+1);
dg(1,1)=0;
theta= pinv((X' * X) + (lambda .* dg)) * (X' * yData);for i=1:m2
    for j=0:n
        XINPUT(i,j+1) = x_vals(i)^j;
    end
end
v = theta' * XINPUT';
```

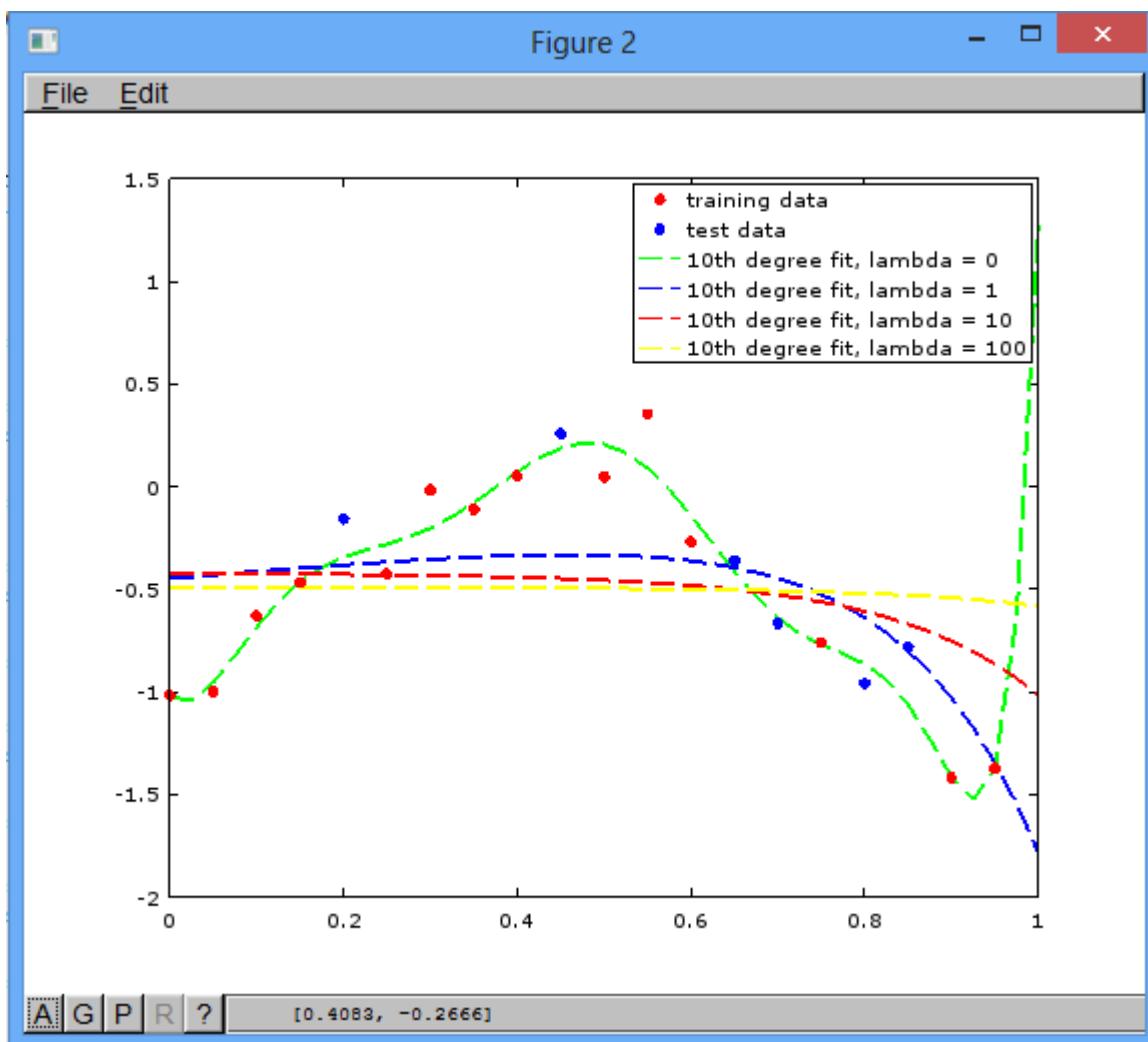
Answer 2(a):

Regression Curve for degree 7 polynomial for lambda = 0, 1, 10, 100



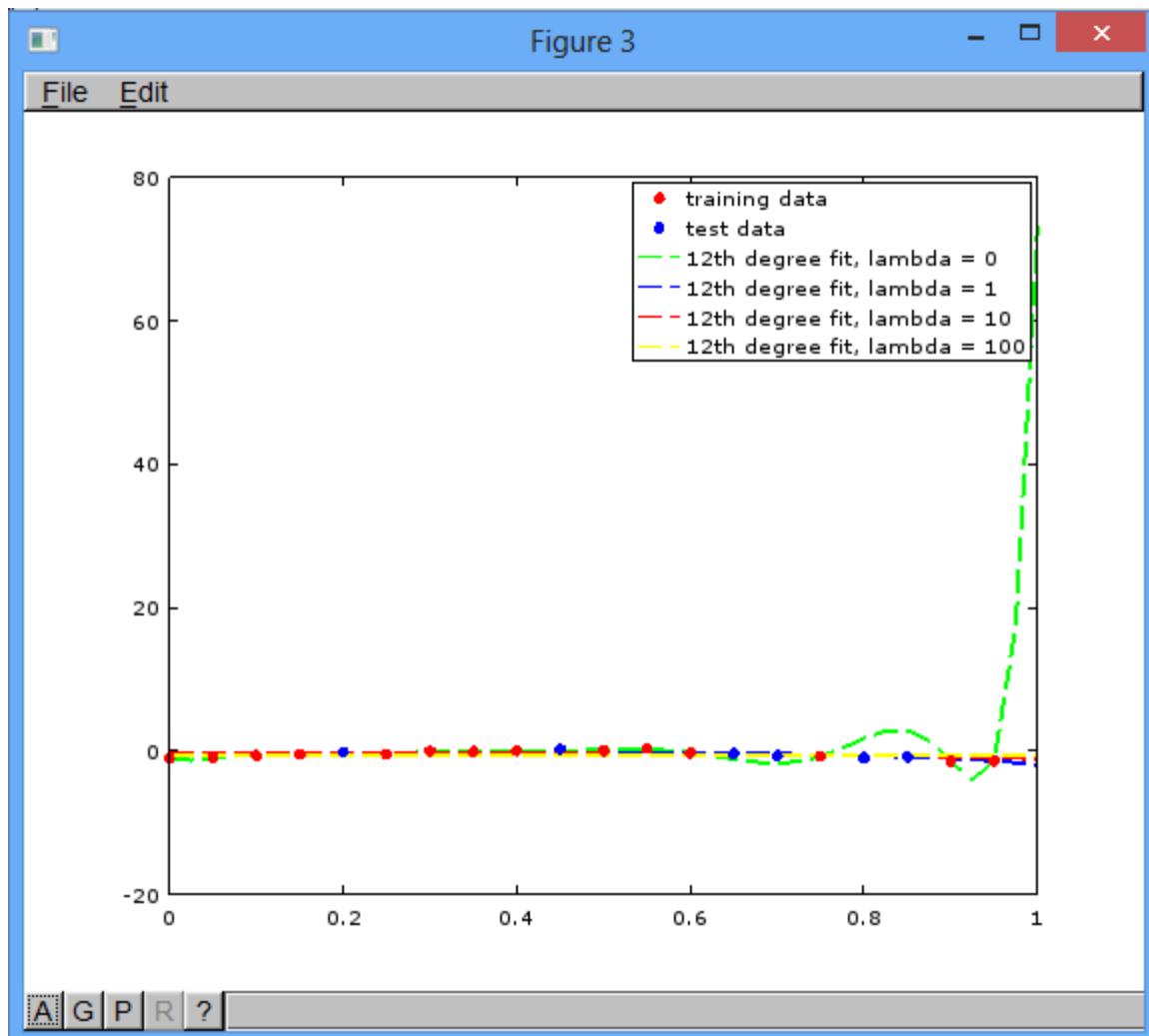
Answer 2(b):

Regression Curve for degree 10 polynomial for lambda = 0, 1, 10, 100



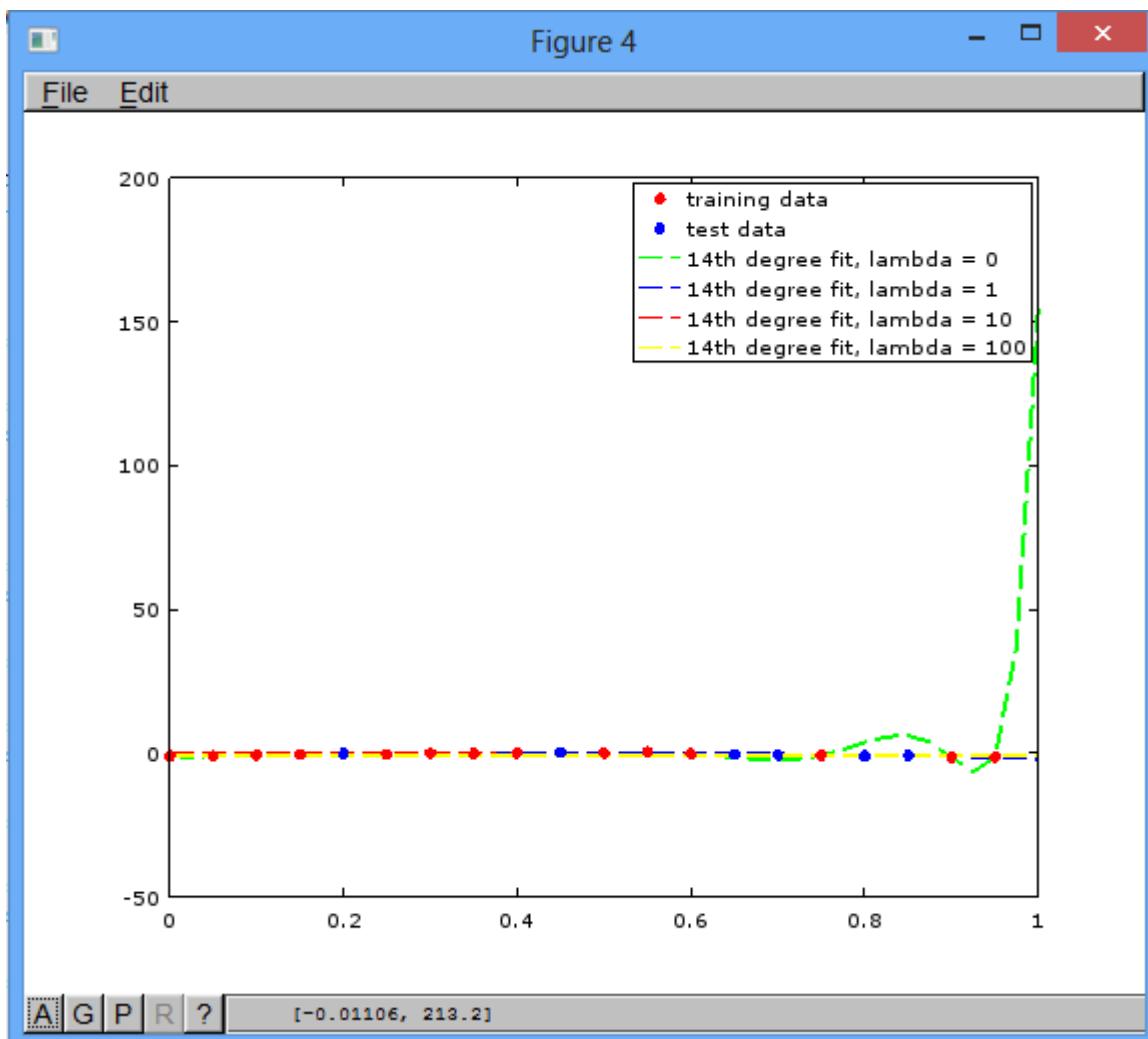
Answer 2(c):

Regression Curve for degree 12 polynomial for lambda = 0, 1, 10, 100



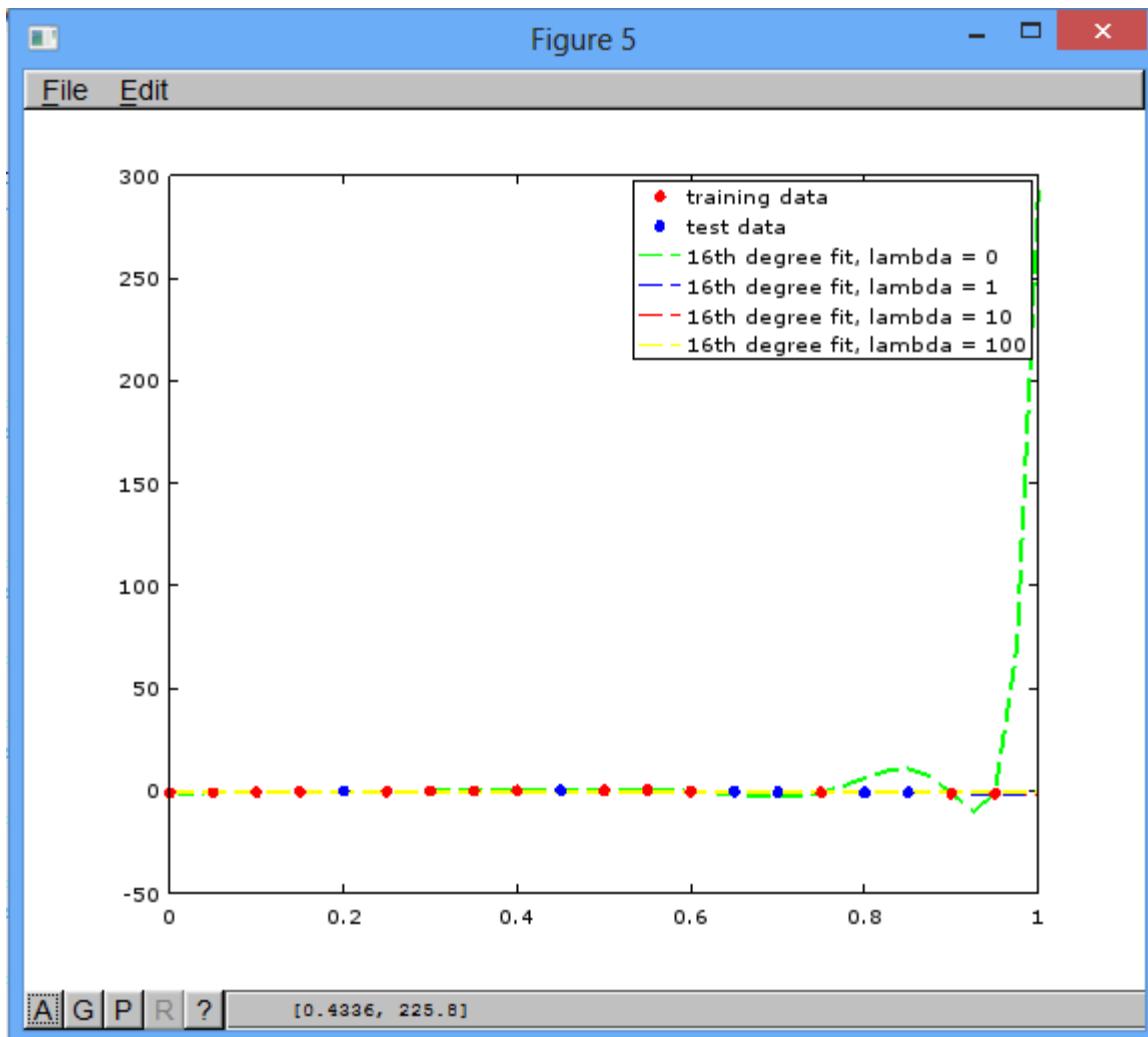
Answer 2(d):

Regression Curve for degree 14 polynomial for lambda = 0, 1, 10, 100



Answer 2(e):

Regression Curve for degree 16 polynomial for lambda = 0, 1, 10, 100



Answer 3:

```
Training and Test error for 7 th degree polynomial

lambda = 0
Training Error = 0.202962      Test Error = 0.393549
lambda = 1
Training Error = 1.955456      Test Error = 0.524491
lambda = 10
Training Error = 3.119545      Test Error = 0.768438
lambda = 100
Training Error = 3.838638      Test Error = 0.986783

Training and Test error for 10 th degree polynomial

lambda = 0
Training Error = 0.172282      Test Error = 0.132000
lambda = 1
Training Error = 1.865078      Test Error = 0.551337
lambda = 10
Training Error = 2.960884      Test Error = 0.748038
lambda = 100
Training Error = 3.795293      Test Error = 0.979479

Training and Test error for 12 th degree polynomial

lambda = 0
Training Error = 0.082681      Test Error = 22.590569
lambda = 1
Training Error = 1.853468      Test Error = 0.569087
lambda = 10
Training Error = 2.901847      Test Error = 0.743822
lambda = 100
Training Error = 3.775901      Test Error = 0.977001

Training and Test error for 14 th degree polynomial

lambda = 0
Training Error = 0.065562      Test Error = 73.723546
lambda = 1
Training Error = 1.852823      Test Error = 0.582507
lambda = 10
Training Error = 2.863166      Test Error = 0.742581
lambda = 100
Training Error = 3.761617      Test Error = 0.975519

Training and Test error for 16 th degree polynomial

lambda = 0
Training Error = 0.054484      Test Error = 189.002381
lambda = 1
Training Error = 1.855594      Test Error = 0.591885
lambda = 10
Training Error = 2.837158      Test Error = 0.742670
lambda = 100
Training Error = 3.750993      Test Error = 0.974618
>> |
```

Answer 4:

Overfitting occurs when the model or the algorithm fits the data too well. Higher the degree of polynomial, more is the overfitting because it can cover most of the training data points in the model.

While applying regularization, as we increase the value of lambda, there will be an increase in the cost function of the training data. For lower degree of polynomial, cost function for test data will also increase with increase in lambda values. For higher degree of polynomial, with increase in lambda values, there will be decrease in the cost function of the test data to a great extent and after that with increase in lambda again the value of cost function for test data will increase gradually.

Patel_Exercises.m

```
training_data=load('trainData.txt');
test_data=load('testData.txt');
training_x=training_data(:,1);
training_y=training_data(:,2);
test_x=test_data(:,1);
test_y=test_data(:,2);
polynomial =[7; 10; 12; 14; 16];
lambda = [0; 1; 10; 100];
x_vals_train = training_x;
x_vals_test = test_x;

figure;
plot(training_x,training_y,'ro','markersize',5,'markerfacecolor','red');
hold on;
plot(test_x,test_y,'bo','markersize',5,'markerfacecolor','blue');
x_vals=(0:0.025:1)';
plot(x_vals,Patel_LinRegRegularized(training_x,training_y,x_vals,7,0),'g'
,'linestyle','--','linewidth',2);
```

```
plot(x_vals,Patel_LinRegRegularized(training_x,training_y,x_vals,7,1),'b','linestyle','--','linewidth',2);

plot(x_vals,Patel_LinRegRegularized(training_x,training_y,x_vals,7,10),'r','linestyle','--','linewidth',2);

plot(x_vals,Patel_LinRegRegularized(training_x,training_y,x_vals,7,100),'y','linestyle','--','linewidth',2);

legend('training data','test data','7th degree fit, lambda = 0','7th degree fit, lambda = 1','7th degree fit, lambda = 10','7th degree fit, lambda = 100','location','southwest');
```

```
figure;
```

```
plot(training_x,training_y,'ro','markersize',5,'markerfacecolor','red');

hold on;

plot(test_x,test_y,'bo','markersize',5,'markerfacecolor','blue');

x_vals=(0:0.025:1);

plot(x_vals,Patel_LinRegRegularized(training_x,training_y,x_vals,10,0),'g','linestyle','--','linewidth',2);

plot(x_vals,Patel_LinRegRegularized(training_x,training_y,x_vals,10,1),'b','linestyle','--','linewidth',2);

plot(x_vals,Patel_LinRegRegularized(training_x,training_y,x_vals,10,10),'r','linestyle','--','linewidth',2);

plot(x_vals,Patel_LinRegRegularized(training_x,training_y,x_vals,10,100),'y','linestyle','--','linewidth',2);
```

```
legend('training data','test data','10th degree fit, lambda = 0','10th
degree fit, lambda = 1','10th degree fit, lambda = 10','10th degree fit,
lambda = 100');

figure;
plot(training_x,training_y,'ro','markersize',5,'markerfacecolor','red');
hold on;
plot(test_x,test_y,'bo','markersize',5,'markerfacecolor','blue');
x_vals=(0:0.025:1);
plot(x_vals,Patel_LinRegRegularized(training_x,training_y,x_vals,12,0),'g','linestyle','--','linewidth',2);
plot(x_vals,Patel_LinRegRegularized(training_x,training_y,x_vals,12,1),'b','linestyle','--','linewidth',2);
plot(x_vals,Patel_LinRegRegularized(training_x,training_y,x_vals,12,10),'r','linestyle','--','linewidth',2);
plot(x_vals,Patel_LinRegRegularized(training_x,training_y,x_vals,12,100),'y','linestyle','--','linewidth',2);
legend('training data','test data','12th degree fit, lambda = 0','12th
degree fit, lambda = 1','12th degree fit, lambda = 10','12th degree fit,
lambda = 100');
```

```
figure;
plot(training_x,training_y,'ro','markersize',5,'markerfacecolor','red');
hold on;
```

```
plot(test_x,test_y,'bo','markersize',5,'markerfacecolor','blue');

x_vals=(0:0.025:1)';

plot(x_vals,Patel_LinRegRegularized(training_x,training_y,x_vals,14,0),'g','linestyle','--','linewidth',2);

plot(x_vals,Patel_LinRegRegularized(training_x,training_y,x_vals,14,1),'b','linestyle','--','linewidth',2);

plot(x_vals,Patel_LinRegRegularized(training_x,training_y,x_vals,14,10),'r','linestyle','--','linewidth',2);

plot(x_vals,Patel_LinRegRegularized(training_x,training_y,x_vals,14,100),'y','linestyle','--','linewidth',2);

legend('training data','test data','14th degree fit, lambda = 0','14th
degree fit, lambda = 1','14th degree fit, lambda = 10','14th degree fit,
lambda = 100');
```

```
figure;

plot(training_x,training_y,'ro','markersize',5,'markerfacecolor','red');

hold on;

plot(test_x,test_y,'bo','markersize',5,'markerfacecolor','blue');

x_vals=(0:0.025:1)';

plot(x_vals,Patel_LinRegRegularized(training_x,training_y,x_vals,16,0),'g','linestyle','--','linewidth',2);

plot(x_vals,Patel_LinRegRegularized(training_x,training_y,x_vals,16,1),'b','linestyle','--','linewidth',2);

plot(x_vals,Patel_LinRegRegularized(training_x,training_y,x_vals,16,10),'r','linestyle','--','linewidth',2);
```

```
plot(x_vals,Patel_LinRegRegularized(training_x,training_y,x_vals,16,100)
),'y','linestyle','--','linewidth',2);

legend('training data','test data','16th degree fit, lambda = 0','16th
degree fit, lambda = 1','16th degree fit, lambda = 10','16th degree fit,
lambda = 100');{}
```

```
for n = 1:length(polynomial)

    printf("\n Training and Test error for %d th degree polynomial
\n\n",polynomial(n));

    for l = 1:length(lambda)

        printf("lambda = %d \n",lambda(l));

        [predict_training] = Patel_LinRegRegularized(training_x, training_y,
x_vals_train,polynomial(n),lambda(l));

        train_error = 0;

        for i = 1:length(x_vals_train)

            train_error = train_error + (training_y(i) - predict_training(i))^2;

        end

        [predict_test] = Patel_LinRegRegularized(training_x, training_y,
x_vals_test,polynomial(n),lambda(l));

        test_error = 0;

        for j = 1:length(x_vals_test)

            test_error = test_error + (test_y(j) - predict_test(j))^2;

        end
```

```
printf("Training Error = %f",train_error);
printf("    Test Error = %f ",test_error);
disp("");
end
end
```