Question 1.

The least squares problem requires the concept of Singular Value Decomposition (SVD). We analyse different solutions. We might get a unique solution, the closest solution, or a set of solutions, from which a smart solution can be used. We discuss the pseudo inverse and how it would help in getting the closest inverse to A, in the case A is not square. After getting the pesudo inverse, we analyse the stability of the pseudo inverse, when a perturbation is introduced while observing y. Applying pseudo inverse to this y will get us a least squares estimate. This is compared to the case when the y is clean, without any noise. The difference between the clean reconstruction and the least squares (with noise) reconstruction (both done using pseudo inverse) is found to be dependent on the singular values of A (which can be obtained through the SVD). We see that the reconstruction can be bad if the smallest singular value is too small. And if the error is additive Gaussian white noise, the average reconstruction error blows up if the smallest singular value is too small. To avoid this, we truncate the terms of the SVD and thus eliminate the possibility of the error term blowing up. The reconstruction error now (between the truncated reconstruction & clean reconstruction) depends on the truncation limit that we have taken (taking this limit depends on the problem we are solving and is mainly intelligent guesswork).

Another type of stable reconstruction is the Tikhonov regularization procedure. This can be interpreted using optimization and it can be computed without the direct computation of the SVD. A new inverse is introduced and we get the noise to be upper bounded. Since the SVD is not explicitly calculated here, computations are cheap even in the case of bigger matrices. In the case of Kernel regression, we proceed by calculating the kernel matrix and using it to develop an inverse (with a delta value dependent on the eigen values of the kernel matrix). This gives us an estimate of the function and we compare it with the original function and estimate errors. We also looked at the different types of kernels used commonly in machine learning. These include polynomial kernels, radial basis kernels and sigmoid kernels. All these methods will work efficiently if the matrix dimensions are considerably small. If not, it would be best to simplify matrices using Cholesky decomposition, QR decomposition, etc. We delve into these topics in the coming classes.

Question 2.g.

Code:

```
clear all;
close all;
A = [1.01 0.99;0.99 0.98];
x = 0;
b = randn(2,10000); % generates a 2x10 matrix with randn values.
for i = 1:10000
    e(:,i) = inv(A)*b(:,i);
    e2(i) = (e(1,i)^2 + e(2,i)^2);
    x = x + e2(i);
end
average = x/10000;
```

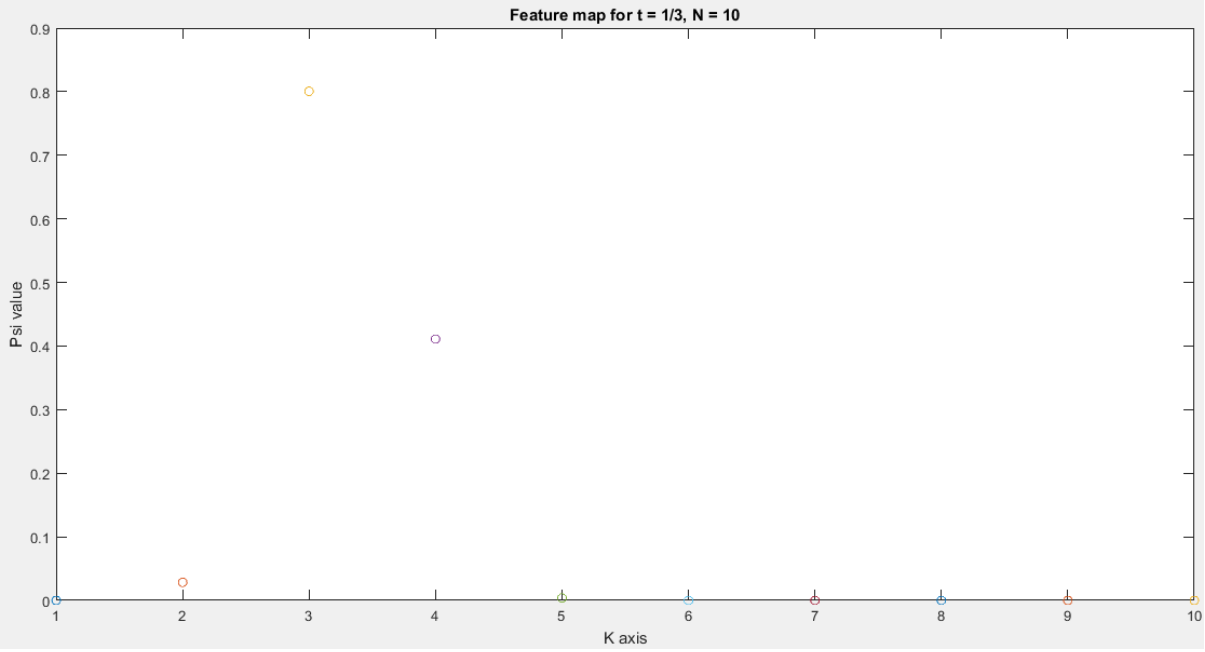
MATLAB output for 'average': (run 3 times since randn function operates randomly each time.)
41687.5607682919
42041.6250830076
418330898.607051

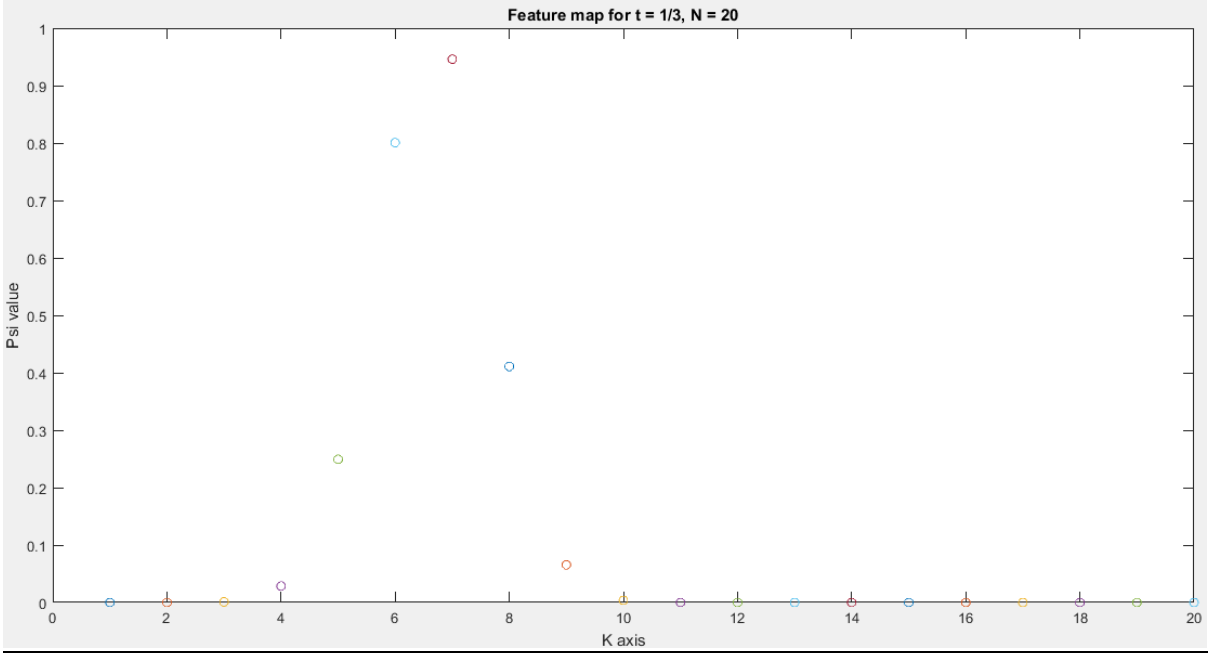
This is almost the same for the (f) part. Thus, verified.

Question 3:

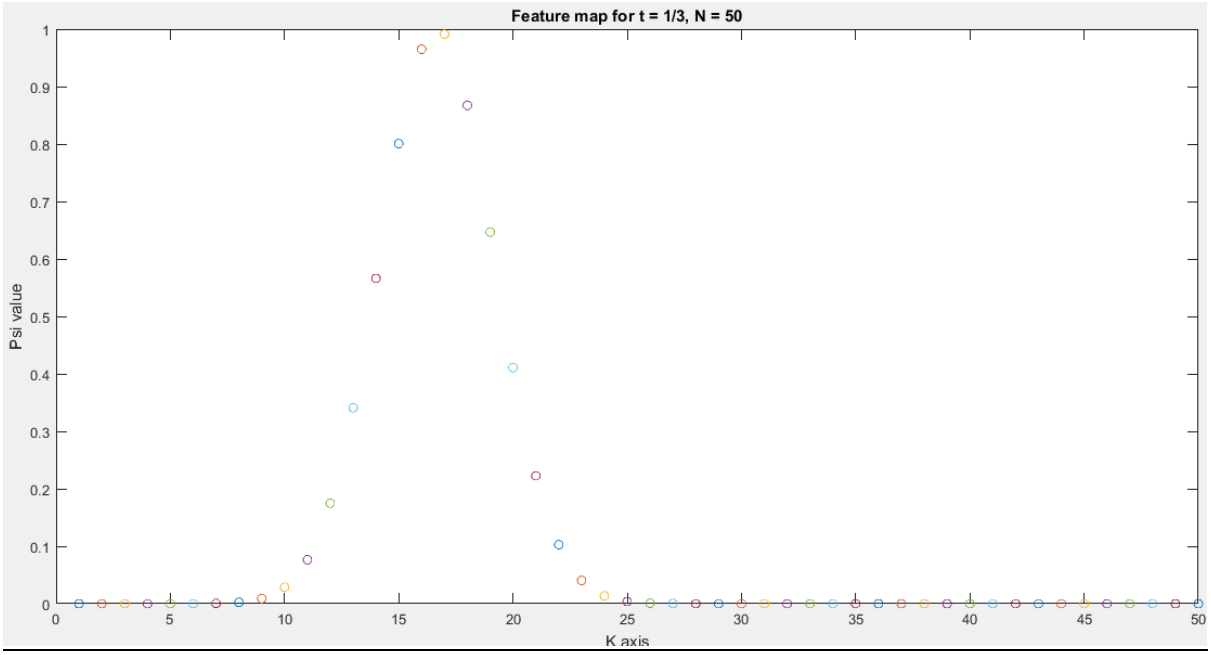
The feature maps have been plotted for each case. The circles are the data points.

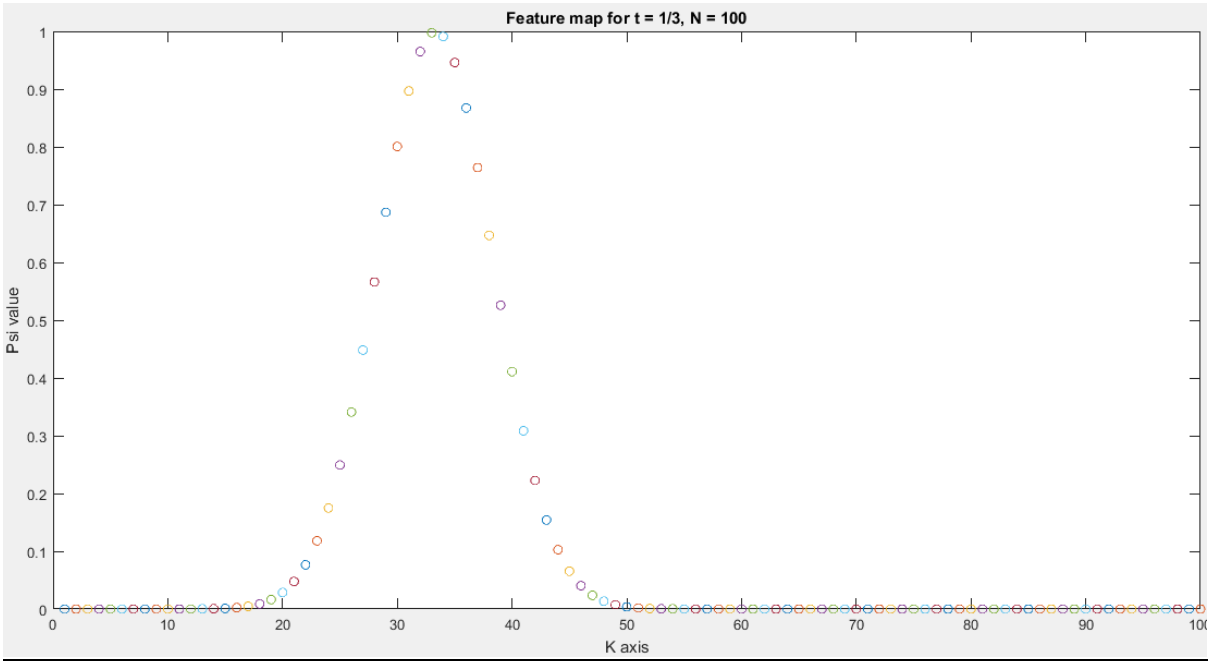
Plots:



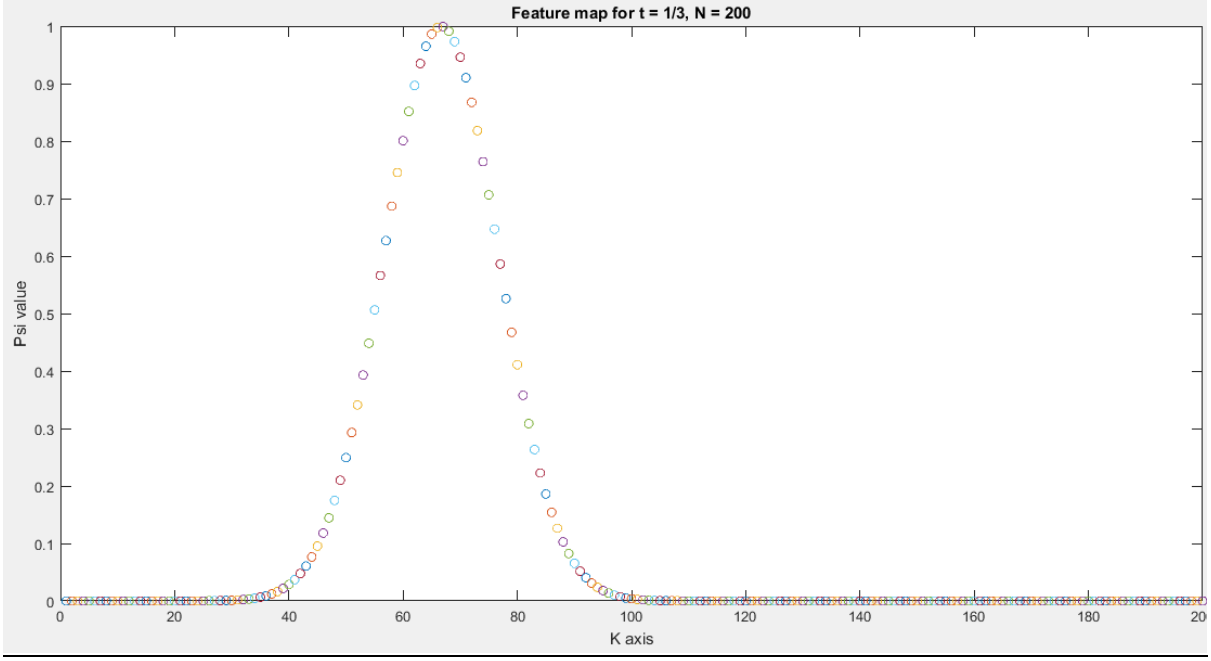


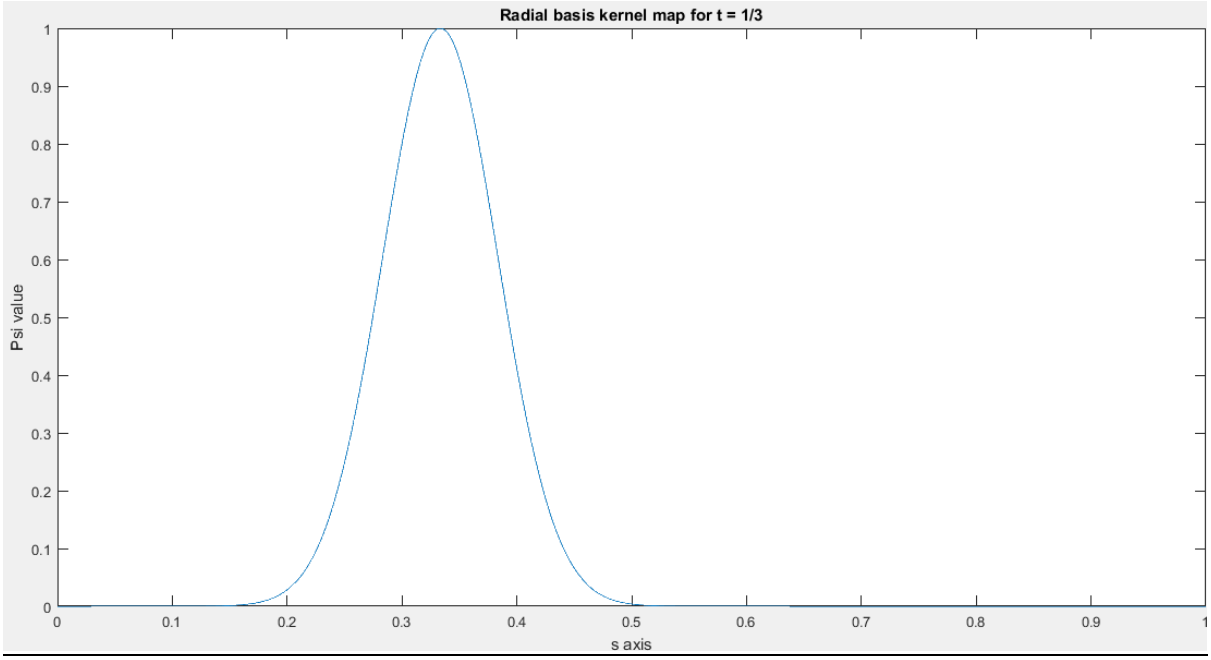
Question 3:


Feature map for t = 1/3, N = 50
Psi value
K axis
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
5
10
15
20
25
30
35
40
45
50


Feature map for t = 1/3, N = 100
Psi value
K axis
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
10
20
30
40
50
60
70
80
90
100

Question 3:





Comments:

In the case of kernel regression, the t_m (in the reconstruction formula) need not be spaced uniformly. They can be at any point as desired. Thus, kernel regression can be changed according to the data that we are trying to fit. But in the non-linear regression case, they are placed in a uniform manner. One can indeed get the non-linear regression basis by setting the kernel basis t_m at uniform intervals. Thus, as we can see from the plots, the bump basis vectors and the radial basis kernel map are of the same form.

Question 3:

Code:

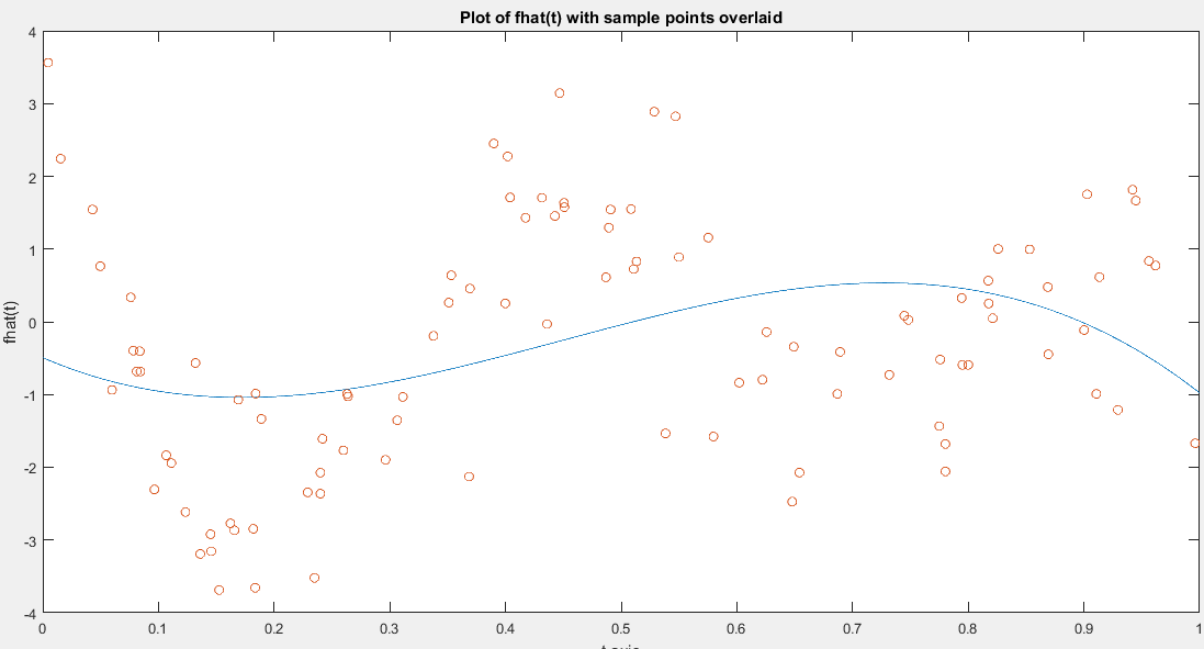
```
clear all;
close all;
t = 1/3;
N = [10, 20, 50, 100, 200];
for i = 1:5
    figure;
    for k = 1:N(i)
    psit(k) = exp(-200*((t-(k/N(i)))^2));
    plot(k, psit(k),'o');
    xlabel ('K axis');
    ylabel ('Psi value');
    hold on;
    end
    title(['Feature map for t = 1/3, N = ' num2str(k) '']);
end
figure;
s = linspace(0,1,5000);
plot(s, exp(-200*((s-(1/3)).^2)));
xlabel ('s axis');
title('Radial basis kernel map for t = 1/3');
ylabel ('Psi value');
```

Question 4.

Codes are provided at the end.

a) Sample error: 15.520228104202692

The best cubic fit to this data:
W3 = -18.3959749927282
W2 = 24.8215751635097
W1 = -6.90972839870063
W0 = -0.492123793899227



b) Generalization error: 1.293859993771946

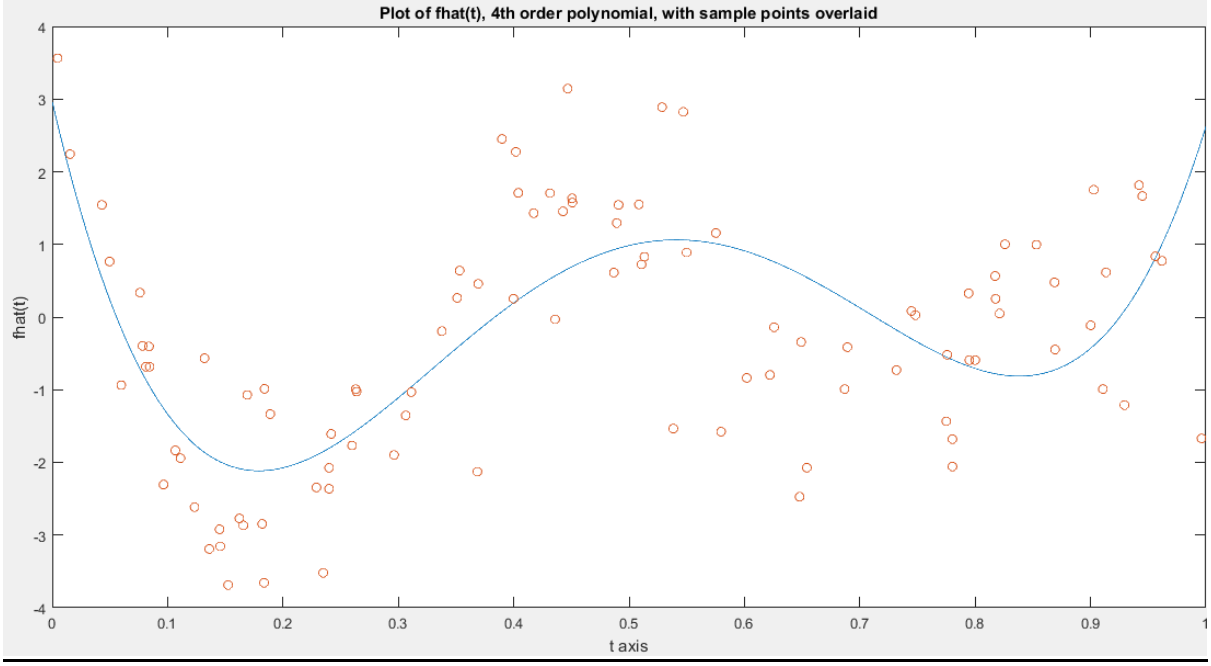
c)

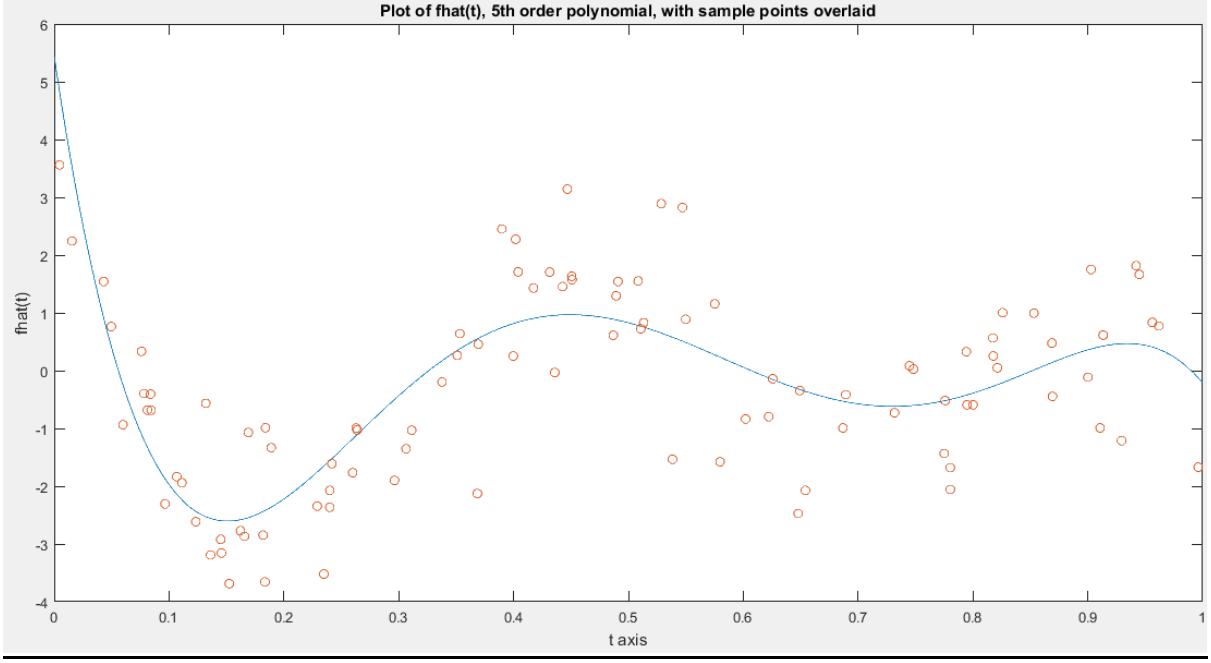
Polynomial of order p.

Order p	Sample error	Generalization error	Largest sing. value of A	Smallest sing. value of A
4	12.1340	0.7725	12.1560	0.00176
5	10.2337	0.4208	12.3196	0.0031
6	9.4812	0.4018	12.4474	5.8977x10^-4
7	9.3540	0.3261	12.5503	1.1005x10^-4
8	9.3167	0.3327	12.6351	1.8578x10^-5
9	9.2539	0.3309	12.7063	3.22x10^-6

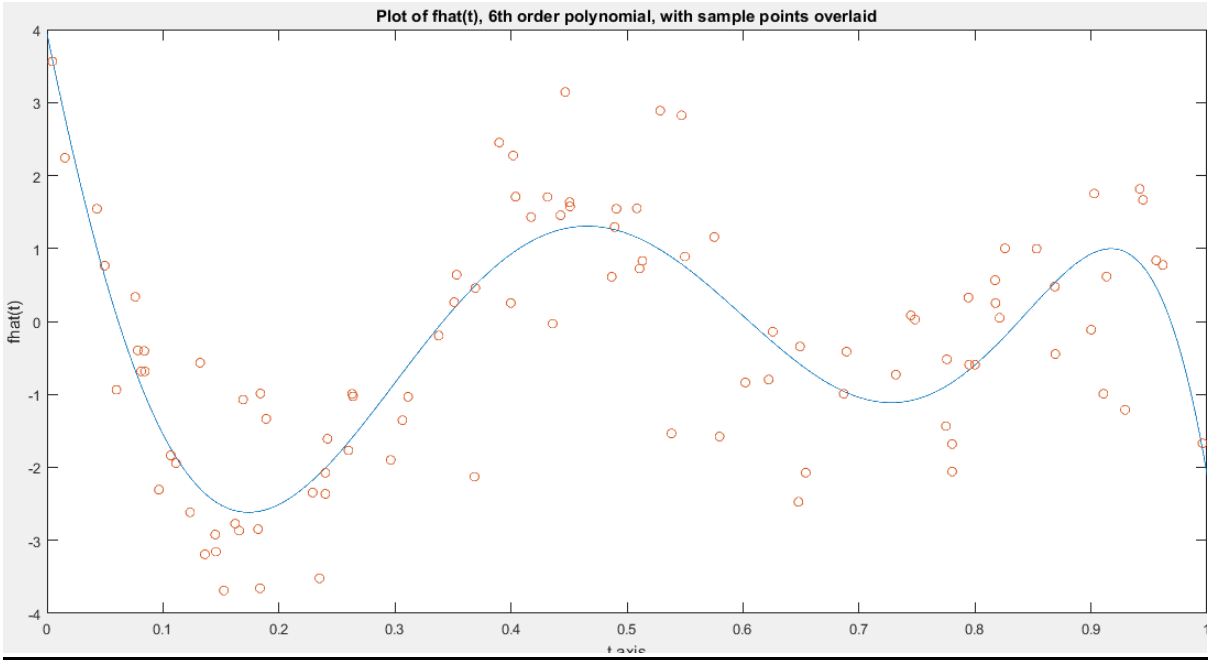
Question 4.

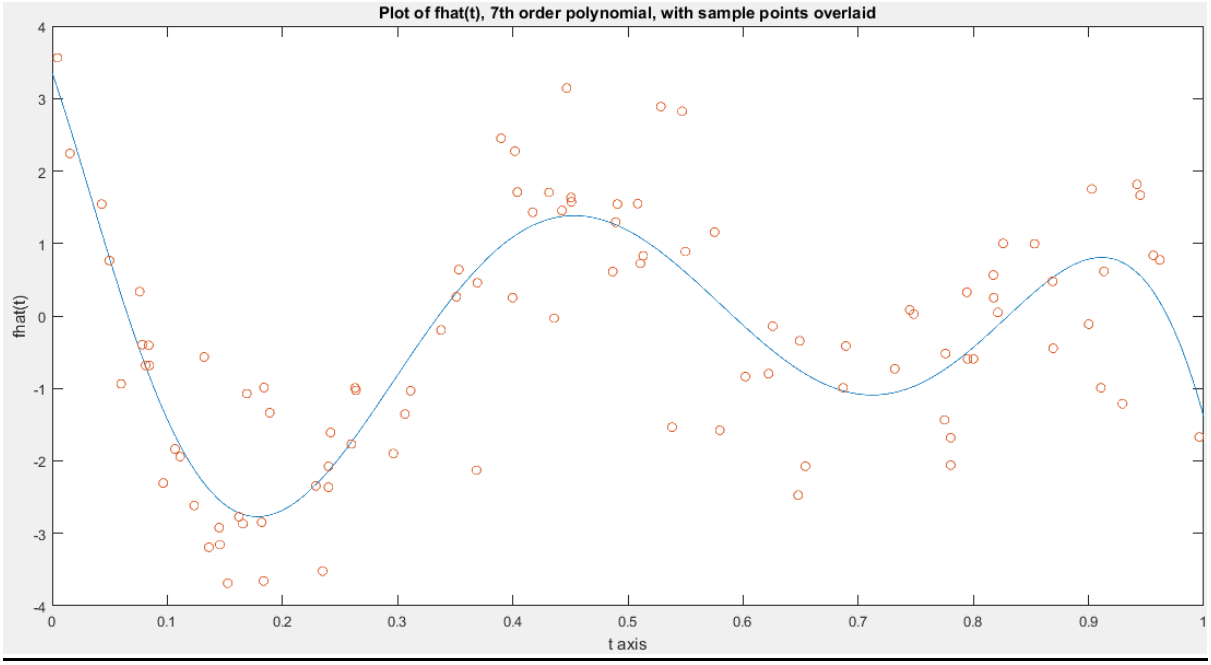
Plots:



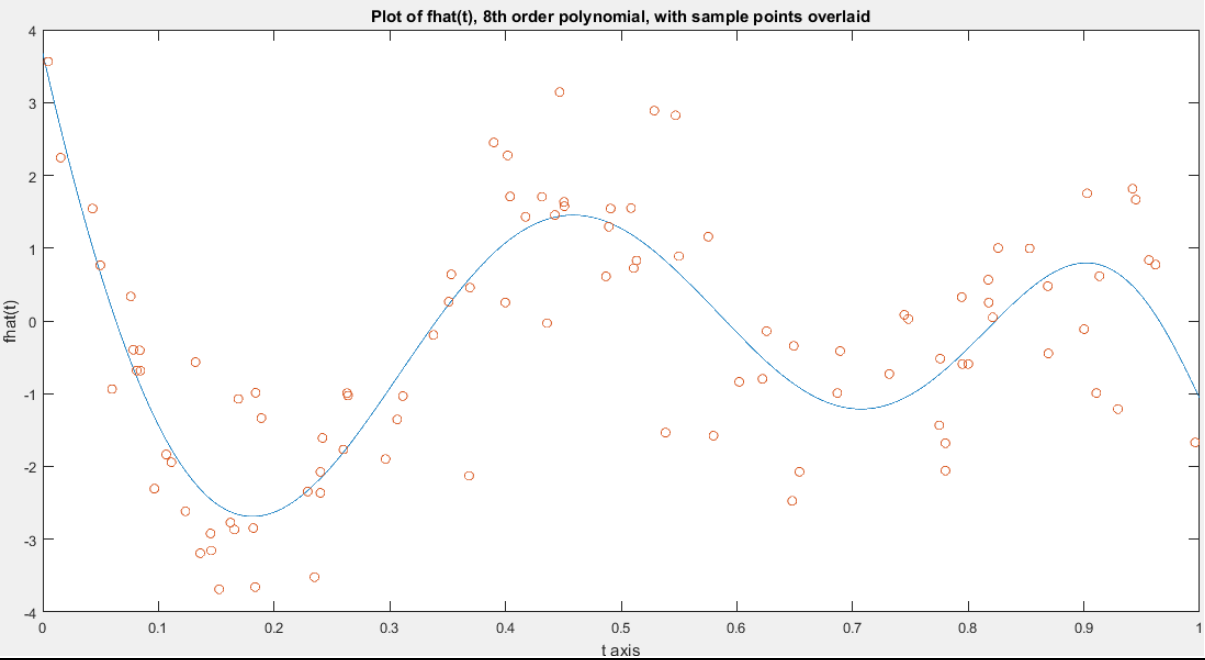


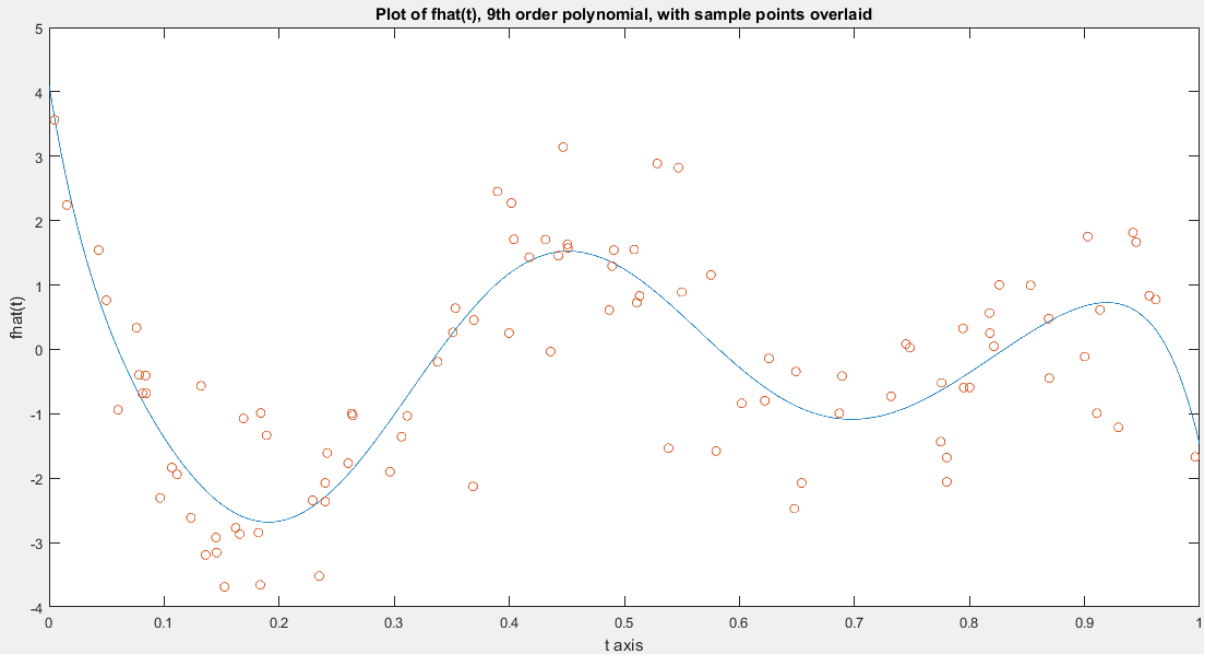
Question 4.


Plot of fhat(t), 6th order polynomial, with sample points overlaid
fhat(t)
t axis


Plot of fhat(t), 7th order polynomial, with sample points overlaid
fhat(t)
t axis

Question 4.





Comments:

For p = 8 and 9, the generalization error increases due to overfitting. But the sample error decreases as expected.

Coefficients of the polynomial in decreasing powers of t:

For p = 9:

-23798.7061897272 (coefficient of t^9)
111140.870240860
-215740.452048344

Question 4.

223833.272265825
-133081.773965033
45550.6656529901
-8918.34362142543
1120.81742236563
-111.928796857435
4.10252141114323

For p = 8:

4486.79069046607
-15941.5379390372
20982.4060728154
-12101.4684617239
2617.74601894856
-167.870806474144
189.170029331071
-69.9739802710004
3.67715612192356

For p = 7:

1987.16478260814
-8234.99009811957
12844.8387557214
-9304.47448018932
2984.68950937058
-236.000470185899
-45.9737692145864
3.35566453150577

For p = 6:

-1308.64088521456
3300.24943619355
-2713.78661916290
603.624430019129
190.089586673773
-77.5780184278458
3.94573775725840

For p = 5:

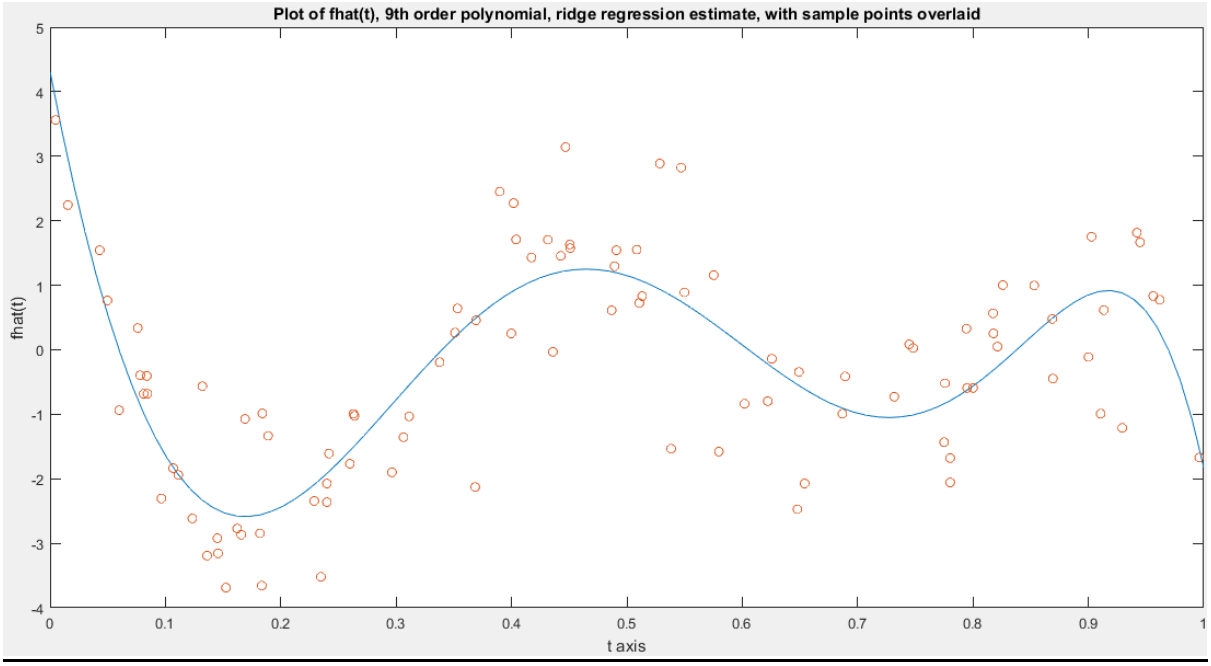
-582.906903286835
1650.04046269520
-1699.03484794543
761.058736321338
-134.810464992548
5.43777459866408

Question 4.

For p = 4:

210.193595834972
-436.753900569838
294.523108705144
-68.3034223143701
2.96351442284634

d)

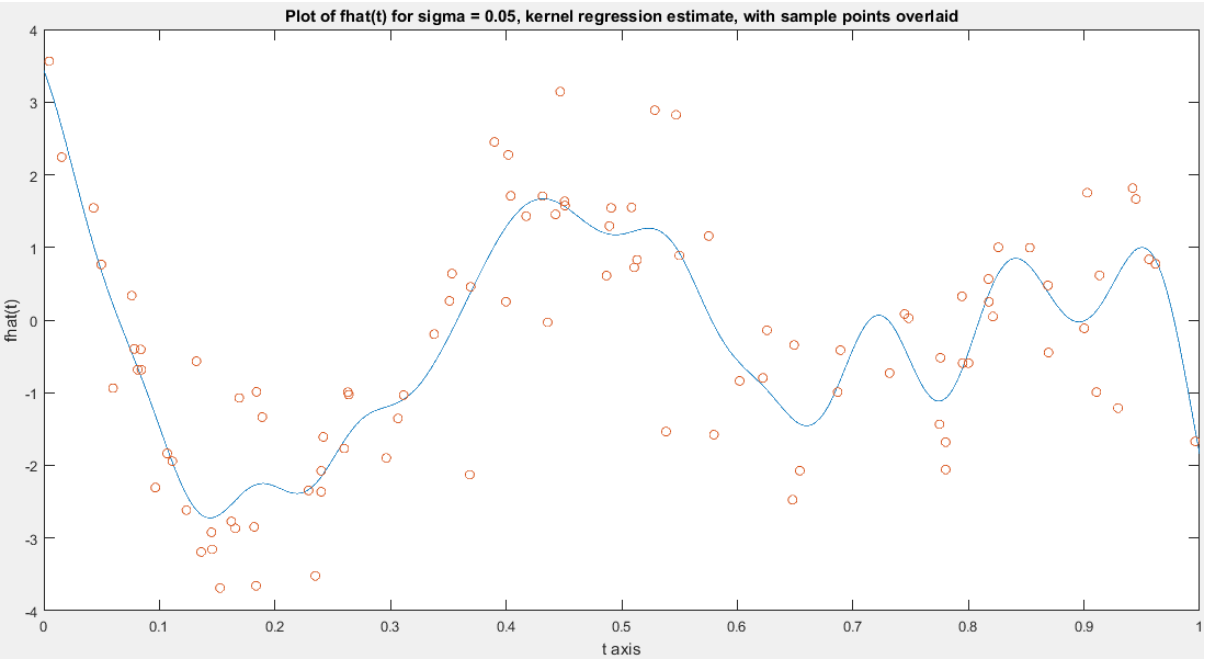


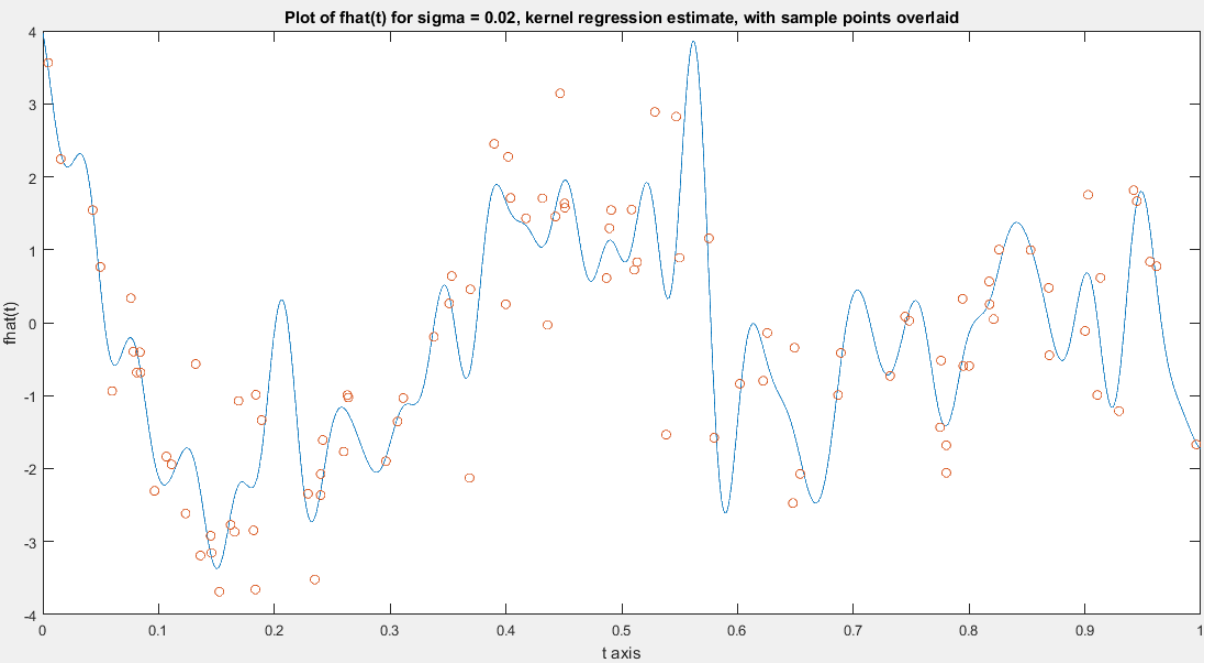
Largest singular value of A: 12.7063
Smallest singular value of A: 3.22x10^-6
Generalization error: 0.3854
Sample error: 9.5167

e)

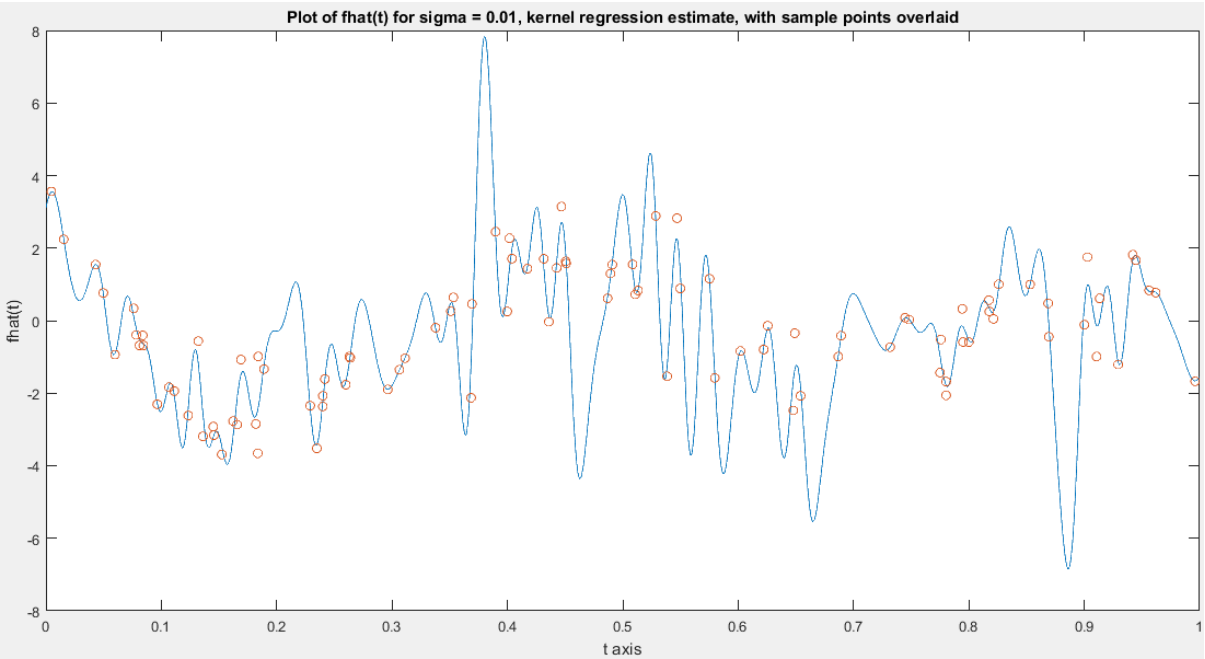
Sigma value	1/20	1/50	1/100	1/300	1/1000
Sample error	8.4528	6.6019	3.8359	1.4968	0.2951
Generalization error	0.5069	0.9975	1.8938	1.3260	1.1644

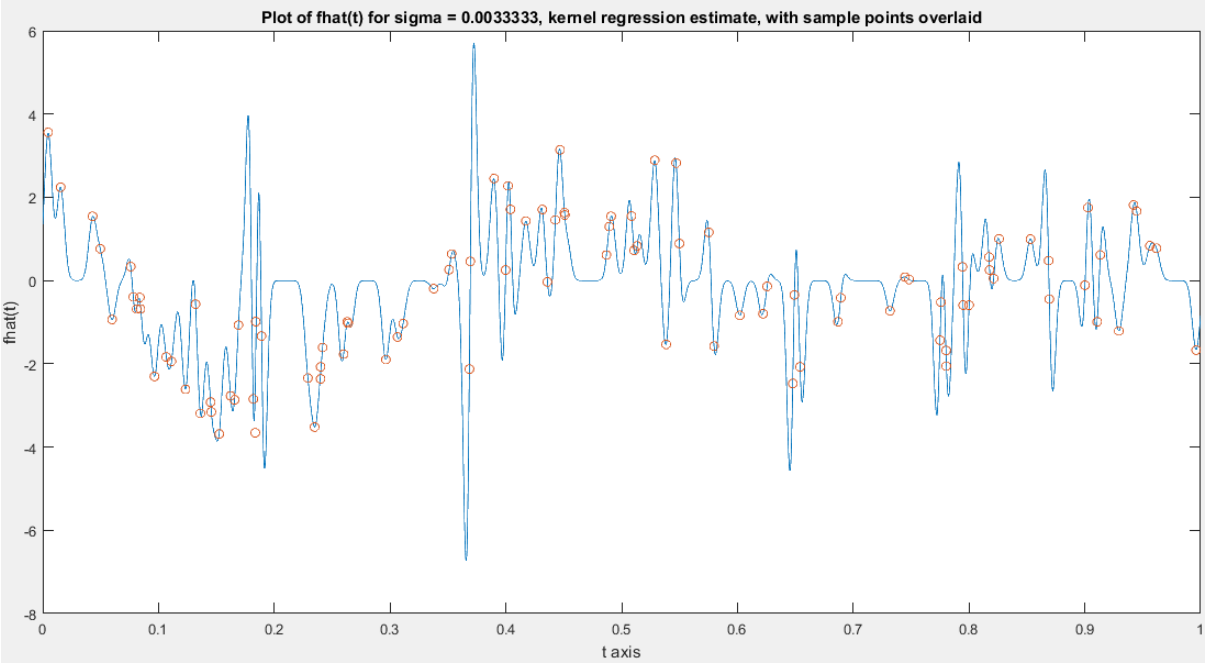
Question 4.


Plot of fhat(t) for sigma = 0.05, kernel regression estimate, with sample points overlaid
fhat(t)
t axis

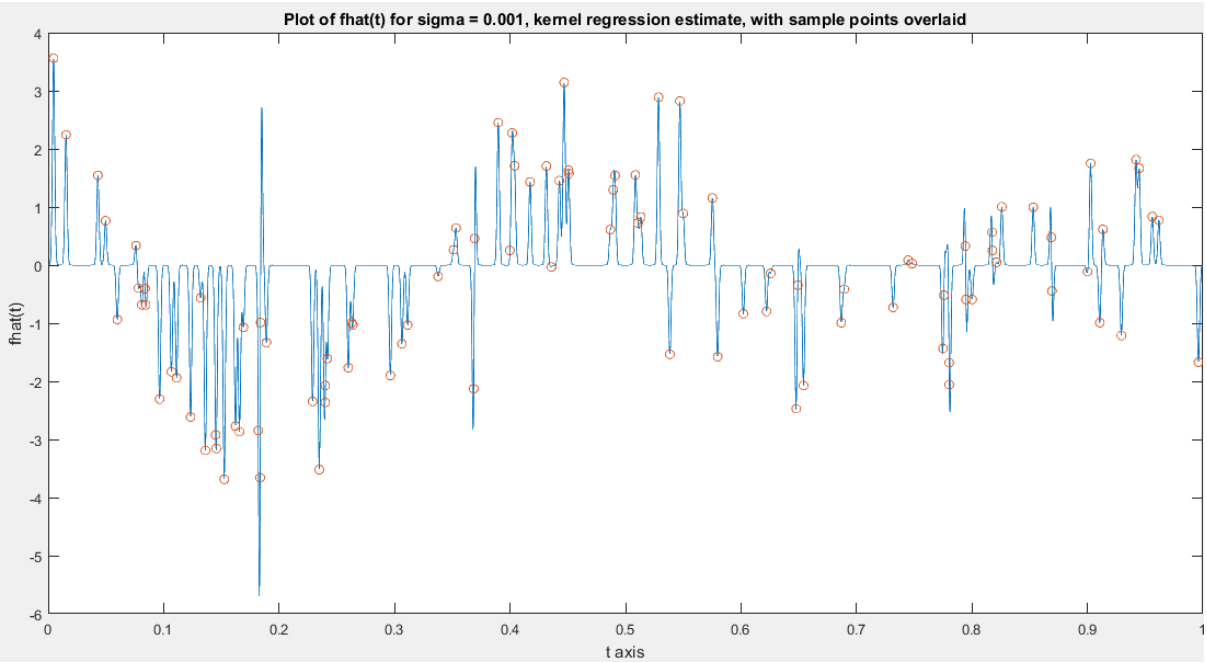

Plot of fhat(t) for sigma = 0.02, kernel regression estimate, with sample points overlaid
fhat(t)
t axis

Question 4.


Plot of fhat(t) for sigma = 0.01, kernel regression estimate, with sample points overlaid
fhat(t)
t axis
8
6
4
2
0
-2
-4
-6
-8
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Plot of fhat(t) for sigma = 0.0033333, kernel regression estimate, with sample points overlaid
fhat(t)
t axis
6
4
2
0
-2
-4
-6
-8
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Question 4.



Comments:

As the plots clearly indicate, for sigma = 1/1000, almost all the points are fit by the curve. This is a case of overfitting. If we observe the generalization errors, we can see that they increase up to sigma = 1/100 and then decrease. So, we have a maximum overfitting at this sigma = 1/100. For very low sigma, the generalization error reduces because the functions become similar to spikes and they approximate a point well. The case of sample error shows that it decreases monotonically, as expected.

Question 4.

CODES:

4.A.

```
clear all
close all
load('hw5p4_data.mat');
for i=1:100
    A(i,:) = [T(i)^3 T(i)^2 T(i) 1]; % ---- A is a 100x4 matrix.
end
B = pinv(A);
x = B*y;
se = 0;
%sample error:
for i=1:100
    se = se + ((y(i) - (x(1)*(T(i)^3) + x(2)*(T(i)^2) + x(3)*(T(i)) +
x(4)))^2);
end
sampleerror = sqrt(se);
t = linspace(0,1,5000);
plot(t,(x(1)*(t.^3) + x(2)*(t.^2) + x(3)*t + x(4)));
ylabel('fhat(t)');
xlabel('t axis');
title('Plot of fhat(t) with sample points overlaid');
hold on;
plot(T,y,'o');
```

4.B.

```
%For 3rd order polynomial
%Sampling 5000 points between 0 and 1 and at intervals of 1/5000.
t = linspace(0,1,5000);
for i = 1:5000
a1(i) = (x(1)*(t(i).^3) + x(2)*(t(i).^2) + x(3)*t(i) + x(4)) -
((sin(12*(t(i) + 0.2)))/(t(i) + 0.2));
b1(i) = a1(i)^2;
end
c1 = mean(b1);
generror3 = c1^(0.5);
```

Question 4.

4.C.

```
%For 4th order polynomial
for i=1:100
    A1(i,:) = [T(i)^4 T(i)^3 T(i)^2 T(i) 1]; % ---- A is a 100x5 matrix.
end
B = pinv(A1);
x = B*y;
se = 0;
%sample error:
for i=1:100
    se(i) = (y(i) - ((x(1)*(T(i)^4) + x(2)*(T(i)^3) + x(3)*(T(i)^2) +
x(4)*(T(i)) + x(5))));
    de(i) = se(i)^2;
end
sampleerrorA1 = sqrt(sum((de)));
figure;
t = linspace(0,1,5000);
plot(t,(x(1)*(t.^4) + x(2)*(t.^3) + x(3)*(t.^2) + x(4)*t + x(5)));
ylabel('fhat(t)');
xlabel('t axis');
title('Plot of fhat(t), 4th order polynomial, with sample points
overlaid');
hold on;
plot(T,y,'o');
s = svd(A1);
smallestsingA1 = min(s);
largestsingA1 = max(s);

t = linspace(0,1,5000);
for i = 1:5000
a2(i) = (x(1)*(t(i).^4) + x(2)*(t(i).^3) + x(3)*(t(i).^2) + x(4)*t(i) +
x(5)) - ((sin(12*(t(i) + 0.2)))/(t(i) + 0.2));
b2(i) = a2(i)^2;
end
c2 = mean(b2);
generror4 = sqrt(c2);

%For 5th order polynomial
for i=1:100
    A2(i,:) = [T(i)^5 T(i)^4 T(i)^3 T(i)^2 T(i) 1]; % ---- A is a 100x6
matrix.
end
B = pinv(A2);
x = B*y;
se = 0;
%sample error:
for i=1:100
    se(i) = (y(i) - ((x(1)*(T(i)^5) + x(2)*(T(i)^4) + x(3)*(T(i)^3) +
x(4)*(T(i)^2) + x(5)*(T(i)) + x(6))));
    de(i) = se(i)^2;
end
sampleerrorA2 = sqrt(sum((de)));
figure;
t = linspace(0,1,5000);
plot(t,((x(1)*(t.^5) + x(2)*(t.^4) + x(3)*(t.^3) + x(4)*(t.^2) + x(5)*(t) +
x(6))));
ylabel('fhat(t)');
xlabel('t axis');
title('Plot of fhat(t), 5th order polynomial, with sample points
overlaid');
```

Question 4.

```
hold on;
plot(T,y,'o');
s = svd(A2);
smallestsingA2 = min(s);
largestsingA2 = max(s);

t = linspace(0,1,5000);
for i = 1:5000
a3(i) = (x(1)*(t(i).^5) + x(2)*(t(i).^4) + x(3)*(t(i).^3) + x(4)*(t(i).^2)
+ x(5)*(t(i)) + x(6)) - ((sin(12*(t(i) + 0.2)))/(t(i) + 0.2));
b3(i) = a3(i)^2;
end
c3 = mean(b3);
generror5 = sqrt(c3);


%For 6th order polynomial
for i=1:100
    A3(i,:) = [T(i)^6 T(i)^5 T(i)^4 T(i)^3 T(i)^2 T(i) 1]; % ---- A is a
100x7 matrix.
end
B = pinv(A3);
x = B*y;
se = 0;
%sample error:
for i=1:100
    se(i) = (y(i) - ((x(1)*(T(i)^6) + x(2)*(T(i)^5) + x(3)*(T(i)^4) +
x(4)*(T(i)^3) + x(5)*(T(i)^2) + x(6)*(T(i)) + x(7))));
    de(i) = se(i)^2;
end
sampleerrorA3 = sqrt(sum((de)));
figure;
t = linspace(0,1,5000);
plot(t,((x(1)*(t.^6) + x(2)*(t.^5) + x(3)*(t.^4) + x(4)*(t.^3) +
x(5)*(t.^2) + x(6)*t + x(7))));
ylabel('fhat(t)');
xlabel('t axis');
title('Plot of fhat(t), 6th order polynomial, with sample points
overlaid');
hold on;
plot(T,y,'o');
s = svd(A3);
smallestsingA3 = min(s);
largestsingA3 = max(s);

t = linspace(0,1,5000);
for i = 1:5000
a4(i) = (x(1)*(t(i).^6) + x(2)*(t(i).^5) + x(3)*(t(i).^4) + x(4)*(t(i).^3)
+ x(5)*(t(i).^2) + x(6)*t(i) + x(7)) - ((sin(12*(t(i) + 0.2)))/(t(i) +
0.2));
b4(i) = a4(i)^2;
end
c4 = mean(b4);
generror6 = sqrt(c4);

%For 7th order polynomial
for i=1:100
    A4(i,:) = [T(i)^7 T(i)^6 T(i)^5 T(i)^4 T(i)^3 T(i)^2 T(i) 1]; % ---- A
is a 100x8 matrix.
end
```

Question 4.

```
B = pinv(A4);
x = B*y;
se = 0;
%sample error:
for i=1:100
    se(i) = (y(i) - ((x(1)*(T(i)^7) + x(2)*(T(i)^6) + x(3)*(T(i)^5) +
x(4)*(T(i)^4) + x(5)*(T(i)^3) + x(6)*(T(i)^2) + x(7)*(T(i)) + x(8))));
    de(i) = se(i)^2;
end
sampleerrorA4 = sqrt(sum((de)));
figure;
t = linspace(0,1,5000);
plot(t,((x(1)*(t.^7) + x(2)*(t.^6) + x(3)*(t.^5) + x(4)*(t.^4) +
x(5)*(t.^3) + x(6)*(t.^2) + x(7)*t + x(8))));
ylabel('fhat(t)');
xlabel('t axis');
title('Plot of fhat(t), 7th order polynomial, with sample points
overlaid');
hold on;
plot(T,y,'o');
s = svd(A4);
smallestsingA4 = min(s);
largestsingA4 = max(s);

t = linspace(0,1,5000);
for i = 1:5000
a5(i) = (x(1)*(t(i).^7) + x(2)*(t(i).^6) + x(3)*(t(i).^5) + x(4)*(t(i).^4)
+ x(5)*(t(i).^3) + x(6)*(t(i).^2) + x(7)*t(i) + x(8)) - ((sin(12*(t(i) +
0.2)))/(t(i) + 0.2));
b5(i) = a5(i)^2;
end
c5 = mean(b5);
generror7 = sqrt(c5);


%For 8th order polynomial
for i=1:100
   A5(i,:) = [T(i)^8 T(i)^7 T(i)^6 T(i)^5 T(i)^4 T(i)^3 T(i)^2 T(i) 1]; % -
--- A is a 100x9 matrix.
end
B = pinv(A5);
x = B*y;
se = 0;
%sample error:
for i=1:100
    se(i) = (y(i) - ((x(1)*(T(i)^8) + x(2)*(T(i)^7) + x(3)*(T(i)^6) +
x(4)*(T(i)^5) + x(5)*(T(i)^4) + x(6)*(T(i)^3) + x(7)*(T(i)^2) + x(8)*(T(i))
+ x(9))));
    de(i) = se(i)^2;
end
sampleerrorA5 = sqrt(sum((de)));
figure;
t = linspace(0,1,5000);
plot(t,((x(1)*(t.^8) + x(2)*(t.^7) + x(3)*(t.^6) + x(4)*(t.^5) +
x(5)*(t.^4) + x(6)*(t.^3) + x(7)*(t.^2) + x(8)*t + x(9))));
ylabel('fhat(t)');
xlabel('t axis');
title('Plot of fhat(t), 8th order polynomial, with sample points
overlaid');
hold on;
plot(T,y,'o');
```

Question 4.

```
s = svd(A5);
smallestsingA5 = min(s);
largestsingA5 = max(s);

t = linspace(0,1,5000);
for i = 1:5000
a6(i) = (x(1)*(t(i).^8) + x(2)*(t(i).^7) + x(3)*(t(i).^6) + x(4)*(t(i).^5)
+ x(5)*(t(i).^4) + x(6)*(t(i).^3) + x(7)*(t(i).^2) + x(8)*t(i) + x(9)) -
((sin(12*(t(i) + 0.2)))/(t(i) + 0.2));
b6(i) = a6(i)^2;
end
c6 = mean(b6);
generror8 = sqrt(c6);

%For 9th order polynomial
for i=1:100
    A6(i,:) = [T(i)^9 T(i)^8 T(i)^7 T(i)^6 T(i)^5 T(i)^4 T(i)^3 T(i)^2 T(i)
1]; % ---- A is a 100x10 matrix.
end
B = pinv(A6);
x = B*y;
se = 0;
%sample error:
for i=1:100
    se(i) = (y(i) - ((x(1)*(T(i)^9) + x(2)*(T(i)^8) + x(3)*(T(i)^7) +
x(4)*(T(i)^6) + x(5)*(T(i)^5) + x(6)*(T(i)^4) + x(7)*(T(i)^3) +
x(8)*(T(i)^2) + x(9)*(T(i)) + x(10))));
    de(i) = se(i)^2;
end
sampleerrorA6 = sqrt(sum((de)));
figure;
t = linspace(0,1,5000);
plot(t,((x(1)*(t.^9) + x(2)*(t.^8) + x(3)*(t.^7) + x(4)*(t.^6) +
x(5)*(t.^5) + x(6)*(t.^4) + x(7)*(t.^3) + x(8)*(t.^2) + x(9)*t + x(10))));
ylabel('fhat(t)');
xlabel('t axis');
title('Plot of fhat(t), 9th order polynomial, with sample points
overlaid');
hold on;
plot(T,y,'o');
s = svd(A6);
smallestsingA6 = min(s);
largestsingA6 = max(s);

t = linspace(0,1,5000);
for i = 1:5000
a7(i) = (x(1)*(t(i).^9) + x(2)*(t(i).^8) + x(3)*(t(i).^7) + x(4)*(t(i).^6)
+ x(5)*(t(i).^5) + x(6)*(t(i).^4) + x(7)*(t(i).^3) + x(8)*(t(i).^2) +
x(9)*t(i) + x(10)) - ((sin(12*(t(i) + 0.2)))/(t(i) + 0.2));
b7(i) = a7(i)^2;
end
c7 = mean(b7);
generror9 = sqrt(c7);
```

Question 4.

4.D.

```
del = 10^(-6)*eye(100);
%For 9th order polynomial
for i=1:100
    Anew(i,:) = [T(i)^9 T(i)^8 T(i)^7 T(i)^6 T(i)^5 T(i)^4 T(i)^3 T(i)^2
T(i) 1]; % ---- Anew is a 100x10 matrix.
end
B = Anew'*inv((Anew*Anew') + del);
x = B*y;
se = 0;
%sample error:
for i=1:100
    se(i) = (y(i) - ((x(1)*(T(i)^9) + x(2)*(T(i)^8) + x(3)*(T(i)^7) +
x(4)*(T(i)^6) + x(5)*(T(i)^5) + x(6)*(T(i)^4) + x(7)*(T(i)^3) +
x(8)*(T(i)^2) + x(9)*(T(i)) + x(10))));
    de(i) = se(i)^2;
end
sampleerrorAnew = sqrt(sum((de)));
figure;
t = linspace(0,1);
plot(t,((x(1)*(t.^9) + x(2)*(t.^8) + x(3)*(t.^7) + x(4)*(t.^6) +
x(5)*(t.^5) + x(6)*(t.^4) + x(7)*(t.^3) + x(8)*(t.^2) + x(9)*t + x(10))));
ylabel('fhat(t)');
xlabel('t axis');
title('Plot of fhat(t), 9th order polynomial, ridge regression estimate,
with sample points overlaid');
hold on;
plot(T,y,'o');
s = svd(Anew);
smallestsingAnew = min(s);
largestsingAnew = max(s);

t = linspace(0,1,5000);
for i = 1:5000
a8(i) = (x(1)*(t(i).^9) + x(2)*(t(i).^8) + x(3)*(t(i).^7) + x(4)*(t(i).^6)
+ x(5)*(t(i).^5) + x(6)*(t(i).^4) + x(7)*(t(i).^3) + x(8)*(t(i).^2) +
x(9)*t(i) + x(10)) - ((sin(12*(t(i) + 0.2)))/(t(i) + 0.2));
b8(i) = a8(i)^2;
end
c8 = mean(b8);
generrord = sqrt(c8);
```

Question 4.

4.E.

```
sigma = [1/20 1/50 1/100 1/300 1/1000];
for k = 1:5

    for i = 1:100
    for j = 1:100
        K(i,j) = exp(-((T(i)-T(j))^2)/(2*sigma(k)*sigma(k)));
    end
    end

e = eig(K);
del(k) = max(e)/1000;
delta = del(k)*eye(100);
B = inv(K + delta);
alpha = B*y;

 for i = 1:100
     ft(i) = 0;
    for j = 1:100
        ft(i) = ft(i) + (alpha(j) * exp(-((T(i)-
T(j))^2)/(2*sigma(k)*sigma(k))));
    end
 end
se = 0;
for i=1:100
    se = se + ((y(i) - ft(i))^2);
end
sampleerror(k) = sqrt(se);

tx = linspace(0,1,5000);
for i = 1:5000
     ft(i) = 0;
    for j = 1:100
        ft(i) = ft(i) + (alpha(j) * exp(-((T(j)-
tx(i))^2)/(2*sigma(k)*sigma(k))));
    end
end

%Sampling 5000 points between 0 and 1 and at intervals of 1/5000.
t = linspace(0,1,5000);
for i = 1:5000
ax = (ft(i) - ((sin(12*(t(i) + 0.2)))/(t(i) + 0.2)));
b(i) = ax^2;
end
c = mean(b);
generror(k) = c^(0.5);

figure;
t = linspace(0,1,5000);
plot(t,ft);
ylabel('fhat(t)');
xlabel('t axis');
title(['Plot of fhat(t) for sigma = ' num2str(sigma(k)) ', kernel
regression estimate, with sample points overlaid']);
hold on;
plot(T,y,'o');
end
```