# LAB 6 - Bias and variance

Here, f(x) = x + 2 *sin (1.5* x) , and x is taken from 0 to 2*pik*/( 1.5*1000), here k is taken as 1000.Normal noise is added having mean =0 and std deviation as sqrt(2).

In [162…

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.linalg as la
import math
import random
from scipy.fftpack import fft,fftfreq
from scipy.linalg import toeplitz
from itertools import count
from sklearn.model_selection import train_test_split
x=np.linspace(0,  (2*np.pi*1000)/(1.5*1000),2500)
fx=x+2*np.sin(x*1.5)
noise=np.random.normal(0,np.sqrt(2),2500)
y=fx+noise
```

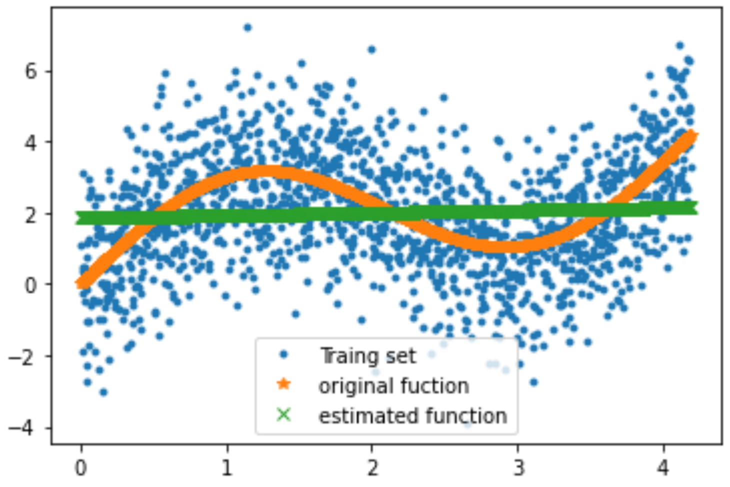
## Question 1:-

For training the model , linear hyposthesis is considered.This , results in underfitting.As shown below, bias is founded as 0.08(high bias) and variance is 0.8.Now,MSE is 2.72.Here MSE will equal to bias^2 + variance + (sigma)^2, where sigma = std deviation of noise.Here , mse will not be equal to bias^2+variance because noise will add extra irreducible error, as due to this original function also will not give MSE =0.As, there will still be error between original and estimated funcitons, overall error will be sum of these 2 errors.Graphs for original and estimated functions are plotted below for both train and test sets.

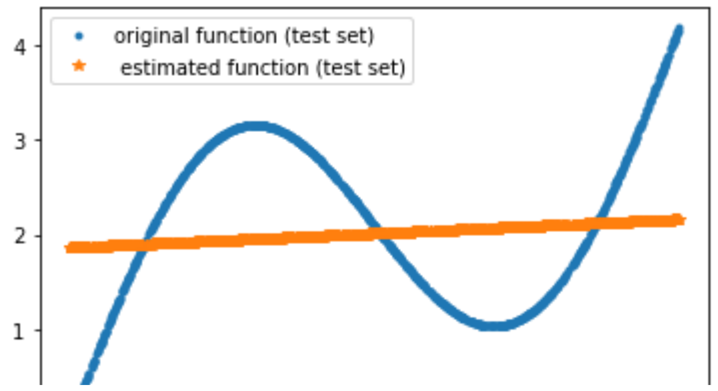
In [178…

```
X=np.c_[ np.ones(len(x)),x]
X_train,X_test,Y_train,Y_test=train_test_split(X,y,test_size=0.4)
XT=X_train.transpose()
temp=np.dot(XT,X_train)
temp=np.linalg.pinv(temp)
temp2=np.dot(XT,Y_train)
theta=np.dot(temp,temp2)
print(str("estimated theta by training the model = "))
print(theta)

fxp=np.matmul(X,theta)
x_train=np.delete(X_train, 0, 1)
x_train=x_train.reshape((x_train.shape[0],))
fx1=x_train+2*np.sin(x_train*1.5)
fxp1=np.matmul(X_train,theta)

plt.plot(x_train,Y_train,'.')
plt.plot(x_train,fx1,'*')
plt.plot(x_train,fxp1,'x')
plt.legend(('Traing set','original fuction ','estimated function'))
plt.show()
x_test=np.delete(X_test, 0, 1)

x_test=x_test.reshape((x_test.shape[0],))

fx2=x_test+2*np.sin(x_test*1.5)
fxp2=np.matmul(X_test,theta)


bias=np.mean(fx2-fxp2)
print(str('Bias = ')+str(bias))
variance=np.var(fx2-fxp2)
mse=np.mean((Y_test-fxp2)**2)
print(str('variance = ')+str(variance))
print(str('MSE = '+str(mse)))

plt.plot(x_test,fx2,'.')
plt.plot(x_test,fxp2,'*')
plt.legend(('original function (test set) ', ' estimated function (test set)'))
plt.show()
```

```
estimated theta by training the model =
[1.8547036  0.07112422]
```



```
Bias = 0.08693735542773387
variance = 0.8080225809894569
MSE = 2.729374922321175
```



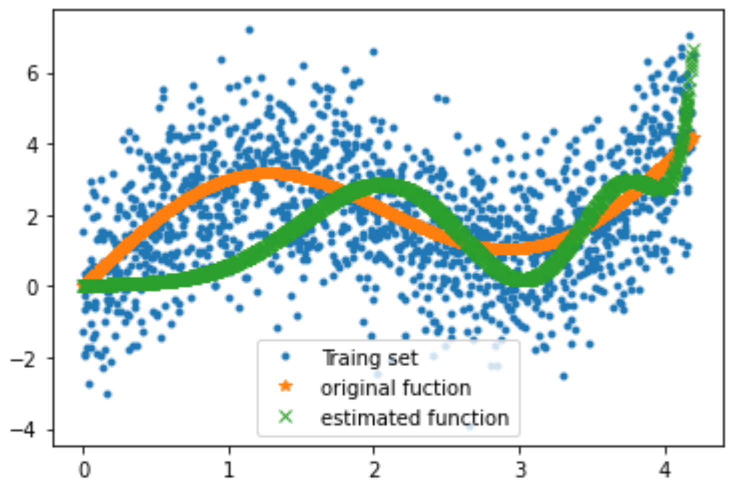
## Question 2:-

For training the model , polynomial of power 12 is considered for hyposthesis.This , results in overfitting.As shown below, bias is founded as 0.6 and variance is 1.1 (high varaiance).Now,MSE is 3.09.Here MSE will equal to bias^2 + variance + (sigma)^2, where sigma = std deviation of noise.Reason for mse not equal to bias^2+ variance is same as earlier question.Graphs for original and estimated functions are plotted below for both train and test sets.

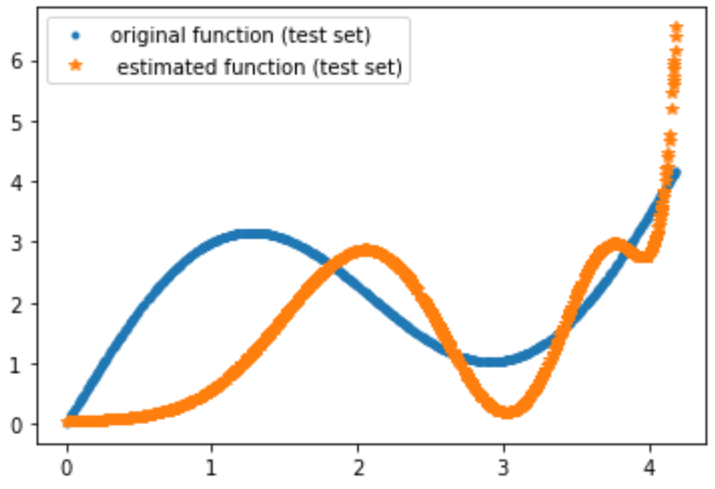
In [177…

```
X=np.c_[ np.ones(len(x)),x,x**2,x**3,x**4,x**5,x**6,x**7,x**9,x**11,x**13]
X_train,X_test,Y_train,Y_test=train_test_split(X,y,test_size=0.4)
XT=X_train.transpose()
temp=np.dot(XT,X_train)
temp=np.linalg.pinv(temp)
temp2=np.dot(XT,Y_train)
theta=np.dot(temp,temp2)
print(str("estimated theta by training the model = "))
print(theta)
fxp=np.matmul(X,theta)
x_train=np.delete(X_train, [0,2,3,4,5,6,7,8,9,10], 1)
x_train=x_train.reshape((x_train.shape[0],))
fx1=x_train+2*np.sin(x_train*1.5)
fxp1=np.matmul(X_train,theta)

plt.plot(x_train,Y_train,'.')
plt.plot(x_train,fx1,'*')
plt.plot(x_train,fxp1,'x')
plt.legend(('Traing set','original fuction ','estimated function'))
plt.show()
x_test=np.delete(X_test, [0,2,3,4,5,6,7,8,9,10], 1)

x_test=x_test.reshape((x_test.shape[0],))

fx2=x_test+2*np.sin(x_test*1.5)
fxp2=np.matmul(X_test,theta)


bias=np.mean(fx2-fxp2)
print(str('Bias = ')+str(bias))
variance=np.var(fx2-fxp2)
mse=np.mean((Y_test-fxp2)**2)
print(str('variance = ')+str(variance))
print(str('MSE = '+str(mse)))

plt.plot(x_test,fx2,'.')
plt.plot(x_test,fxp2,'*')
plt.legend(('original function (test set) ', ' estimated function (test set)'))
plt.show()
```

```
estimated theta by training the model =
[ 3.70948354e-02  5.53034580e-02  8.32447148e-02  1.20196108e-01
  1.56631874e-01  1.62846526e-01  8.06232306e-02 -1.43126444e-01
  1.66834427e-02 -8.94062103e-04  1.75522639e-05]
```



```
Bias = 0.6031331813131114
variance = 1.1009140825739576
MSE = 3.0956251311856064
```

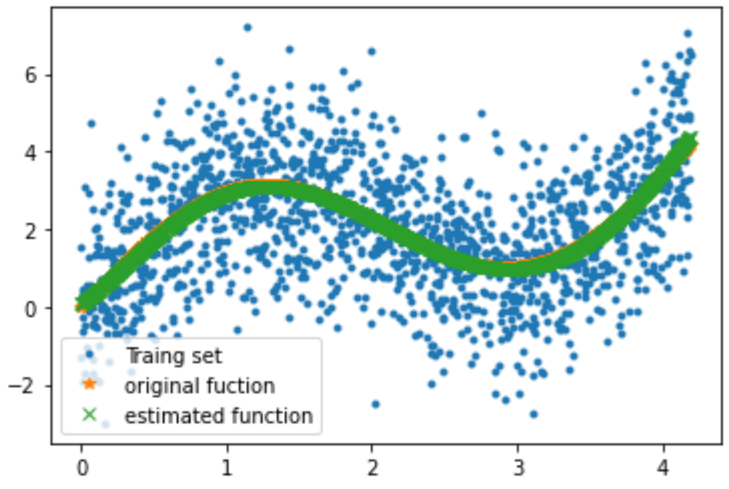


For training the model , polynomial of power 7 is considered as hypothesis.This results in almost perfect estimation.As shown below, bias is founded as 0.05(lower than underfitting) and variance is 0.0029(lower than overfitting).Now,MSE is 1.988~2.As irreducible error is already present having magnitude 2, we can deduce that the mse between original and estimated is approximately 0.Thus, our estimated funciton resembles the original function.Graphs for original and estimated functions are plotted below for both train and test sets.

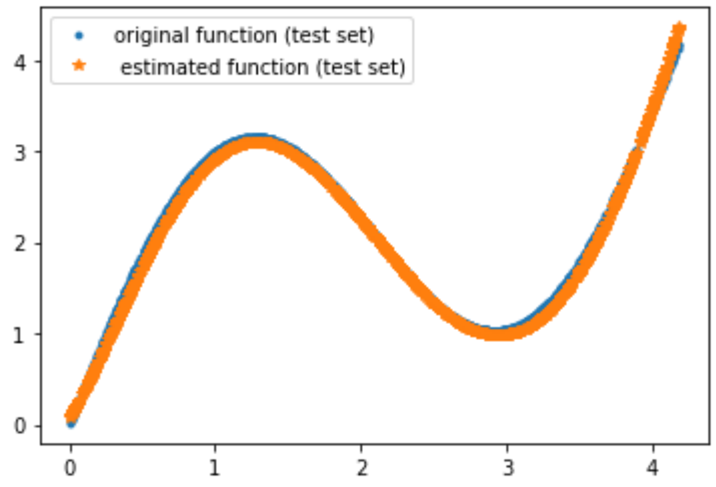
In [176…

```
X=np.c_[ np.ones(len(x)),x,x**2,x**3,x**4,x**5,x**7]
X_train,X_test,Y_train,Y_test=train_test_split(X,y,test_size=0.4)
XT=X_train.transpose()
temp=np.dot(XT,X_train)
temp=np.linalg.pinv(temp)
temp2=np.dot(XT,Y_train)
theta=np.dot(temp,temp2)
print(str("estimated theta by training the model = "))
print(theta)
fxp=np.matmul(X,theta)
x_train=np.delete(X_train, [0,2,3,4,5,6], 1)
x_train=x_train.reshape((x_train.shape[0],))
fx1=x_train+2*np.sin(x_train*1.5)
fxp1=np.matmul(X_train,theta)

plt.plot(x_train,Y_train,'.')
plt.plot(x_train,fx1,'*')
plt.plot(x_train,fxp1,'x')
plt.legend(('Traing set','original fuction ','estimated function'))
plt.show()
x_test=np.delete(X_test, [0,2,3,4,5,6], 1)

x_test=x_test.reshape((x_test.shape[0],))

fx2=x_test+2*np.sin(x_test*1.5)
fxp2=np.matmul(X_test,theta)

bias=np.mean(fx2-fxp2)
print(str('Bias = ')+str(bias))
variance=np.var(fx2-fxp2)
mse=np.mean((Y_test-fxp2)**2)
print(str('variance = ')+str(variance))
print(str('MSE = '+str(mse)))

plt.plot(x_test,fx2,'.')
plt.plot(x_test,fxp2,'*')
plt.legend(('original function (test set) ', ' estimated function (test set)'))
plt.show()
```

```
estimated theta by training the model =
[ 7.73871786e-02  2.60245074e+00  3.04435218e+00 -4.01679566e+00
  1.36504174e+00 -1.56548991e-01  7.44802120e-04]
```



```
Bias = 0.05884514251437038
variance = 0.002911644098327173
MSE = 1.9888155722964895
```


original function (test set)
estimated function (test set)