1 Problem 1

1.1 a

$(Ax)^T b = x^T A^T b = x^T (A^T A\hat{x}) = (x^T A^T)(A\hat{x}) = (Ax)^T (A\hat{x})$

1.2 b

Using part a, we can substitute x for $\hat{x}$, giving us $(A\hat{x})^T b = \hat{x}^T A^T b = ||A\hat{x}||^2$. $\frac{||A\hat{x}||^2}{||Ax||} = ||Ax||||b||$, and therefore, $\frac{||A\hat{x}||^2}{||Ax||||b||} = ||Ax||$. This means the equality holds as the two sides are equivalent to $\frac{||A\hat{x}||^2}{||Ax||||b||}$.

1.3 c

Since we know that $cos(\theta_{\hat{x}}) = \frac{(A\hat{x})^T b}{||A\hat{x}||||b||}$, where $\theta_{\hat{x}}$ is the angle between $A\hat{x}$ and b. The same applies for $cos(\theta_x) = \frac{(Ax)^T b}{||Ax||||b||}$, where θ_x is the angle between Ax and b. We know that $\theta_{\hat{x}} < \theta_x$ because TODO

2 Problem 2

2.1 a

$\hat{x} = (A^T A)^{-1} A^T b = ((QR)^T QR)^{-1} (QR)^T b = (R^T Q^T QR)^{-1} R^T Q^T b = R^{-1} Q^{-1} (Q^T)^{-1} (R^T)^{-1} R^T Q^T b = R^{-1} (Q^T Q)^{-1} Q^T b = R^{-1} Q^T b$, so $A\hat{x} = QRR^{-1} Q^T b = QQ^T b$

2.2 b

$A\hat{x} = A(A^T A)^{-1} A^T b = AA^{-1} (A^T)^{-1} A^T b = b$, so $||A\hat{x} - b||^2 = ||b||^2 + ||A\hat{x}||^2 - 2||A\hat{x}||b = ||b||^2 + ||A\hat{x}||^2 - 2||A\hat{x}||^2 = ||b||^2 - ||A\hat{x}||^2$. From part a, we know that this is equal to $||b||^2 - ||QQ^T b||^2 = ||b||^2 - ||Q^T b||^2$.

3 Problem 3

3.1 a

The weighted least squares is equal to $\sum_{i=1}^{m} w_i (\hat{a_i}^T x - b_i)^2$. The one-norm for vector x is equal to $\sum_{i=1}^{m} |x_i| = 1$, so we can express the weighted sum as the one-norm squared of $\sum_{i=1}^{m} (w_i (\hat{a_i}^T x - b_i))^2$, meaning if we have some diagonal matrix D with entries $\sqrt{w_1}, \sqrt{w_2} ... \sqrt{w_m}$, then this is equal to $||D(Ax - b)||^2$.

3.2 b

For matrix B where $B = DA$, if A has linearly independent columns, then since D is a diagonal matrix, it also has linearly independent columns no w_i will be equal to 0. Therefore, if $Ax = 0$, then x must be 0 as it is not in the column

space of A. This means B is the product of 2 linearly independent matrices, so if $Bx = 0$, $DAx = 0$, and since D is also linearly independent, x must be 0. This means that if $Bx = 0$ only if x is 0, then B must be linearly independent.

3.3 c

We said that $B = DA$, so we can use matrix B in this weighted least squares solution, meaning that since $d = Db$, $\hat{d} = (B^T B)^{-1} B^T d = ((DA)^T DA)^{-1}(DA)^T Db = (A^T D^T DA)^{-1} A^T D^T Db = (A^T diag(w)A)^{-1} A^T diag(w)b = (A^T diag(w)A)^{-1} A^T diag(w)b$

4 Problem 4

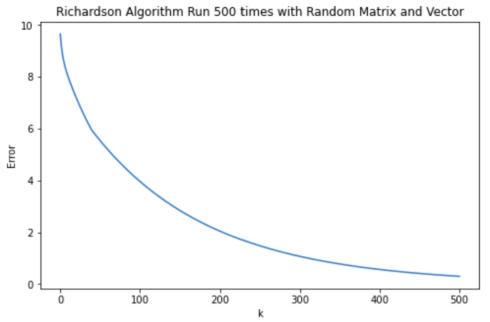
4.1 a

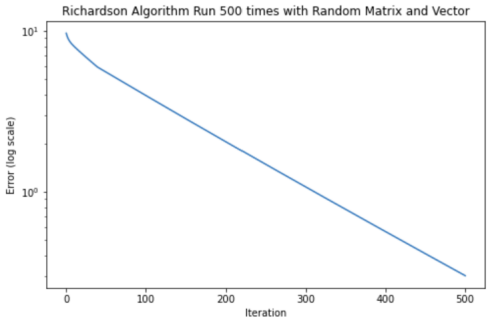
If $x^{k+1} = x^k$, then that means $\mu A^T(Ax^k - b) = 0$. $\hat{x}$ is the solution of $A^T(Ax_k - b) = 0$. This means that $Ax_k = b$, but since all $x_{k+1} = x_k$, all $x_k = 0$. Therefore, $b = 0$, but since $\hat{x} = A^\dagger b$, $\hat{x} = 0$. Therefore, all $x_k = \hat{x}$.

4.2 b

$x^{k+1} = x^k - \mu A^T(Ax^k - b) = x^k - \mu A^T Ax^k + \mu A^T b = x^k(I - \mu A^T A) + \mu A^T b$. Therefore, $x^{k+1} = Fx^k + g$ where $F = (I - \mu A^T A)$ and $g = \mu A^T b$.

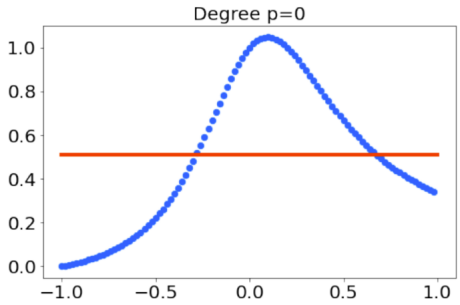
4.3 c

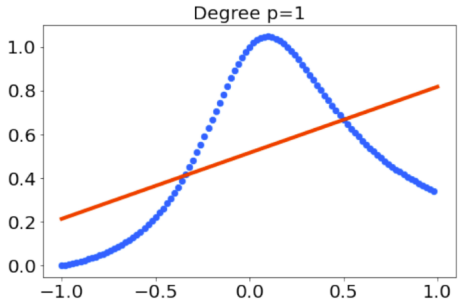


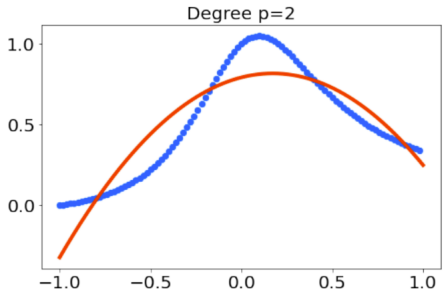


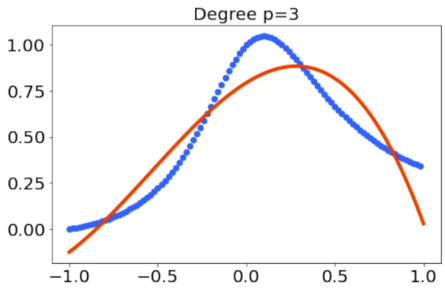
5 Problem 5

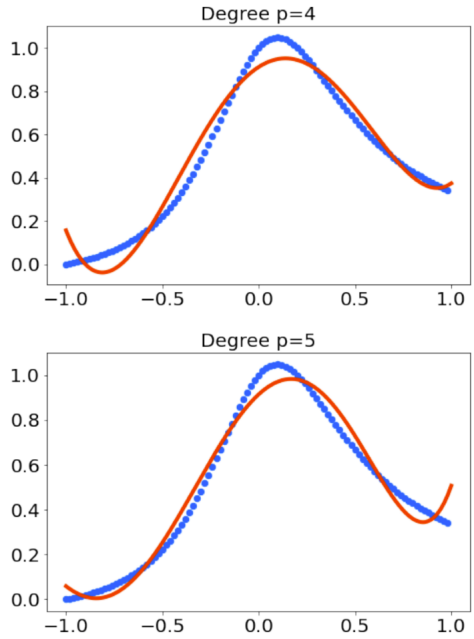
5.1 a



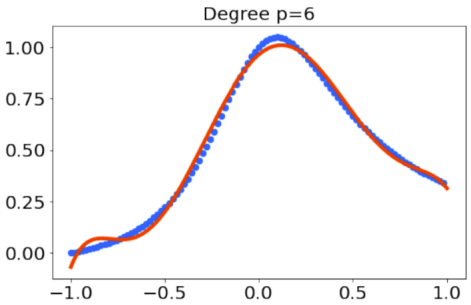


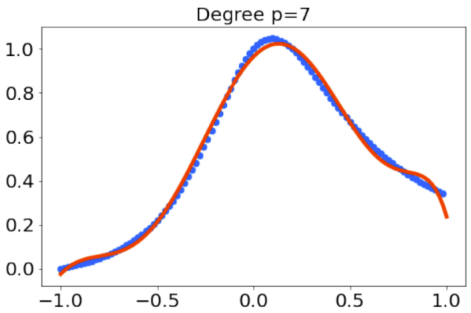


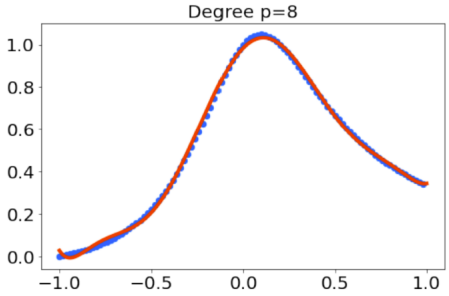



Degree p=4
1.0
0.8
0.6
0.4
0.2
0.0
−1.0
−0.5
0.0
0.5
1.0

Degree p=5
1.0
0.8
0.6
0.4
0.2
0.0
−1.0
−0.5
0.0
0.5
1.0

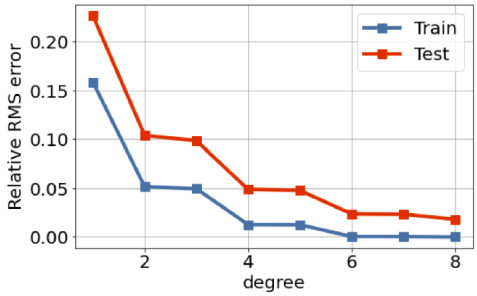

Degree p=6
1.00
0.75
0.50
0.25
0.00
−1.0
−0.5
0.0
0.5
1.0


Degree p=7
1.0
0.8
0.6
0.4
0.2
0.0
−1.0
−0.5
0.0
0.5
1.0



5.2 b

This tells us the reasonable degree fit is around 4-6, as the least difference in degree between train and test occurs among those values



6 Problem 6

$||AX - Y||^2 = ||X^T A^T - Y^T||^2 = ||X^T W - Y^T||^2$, meaning this is also equal to $\sum_{i=1}^{m} ||X^T w_i - y_i||^2$, where w_i and y_i are the columns of the A^T and Y^T matrices, respectively. This means to minimize the summation, we need $X^T w_i = y_i$, or $w_i = (X^T)^{-1} y_i$. Since $W = A^T$, this means $A = W^T$, so $A = ((X^T)^{-1} Y^T)^T$. Since $X^{-1}X = I$, we can say $A = ((X^T)^{-1} X^{-1} X Y^T)^T = ((XX^T)^{-1} X Y^T)^T = YX^T((XX^T)^T)^{-1} = YX^T(X^T X)^{-1} = YX^\dagger$

7 Problem 7

7.1 a

If we let Y be a matrix of N rows and K columns that is populated with 1s and -1s. For any index Y_{nk}, if the value at $X^T B_n$ is in the group k, then $Y_{nk} = 1$. Otherwise, $Y_{nk} = -1$.

7.2 b

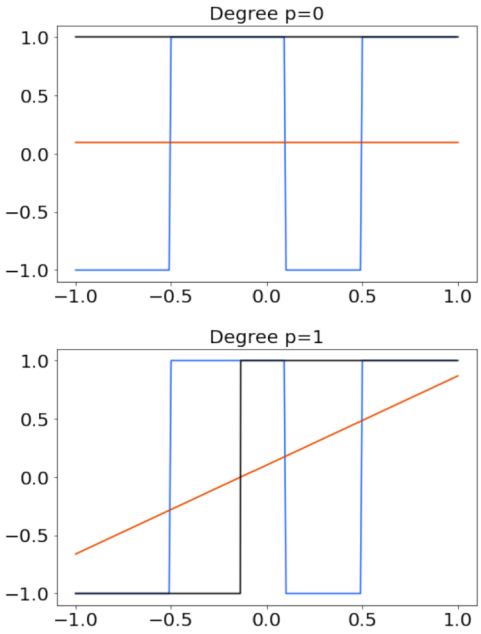
In order to minimize this, we need to set $Y = X^T\beta$, or ensure that for every column k in β and Y, $X^T\beta_k = Y_k$. By taking the norm of the solution, we see that this is then equal to $||X^T\beta_1 - Y_1||^2 + ||X^T\beta_2 - Y_2||^2 ... + ||X^T\beta_k - Y_k||^2$.

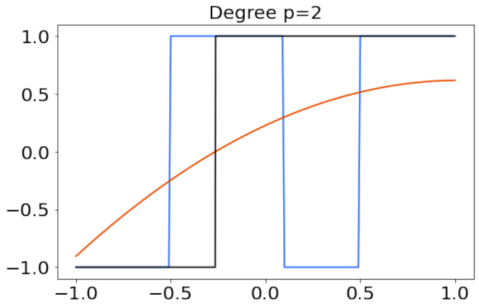
7.3 c

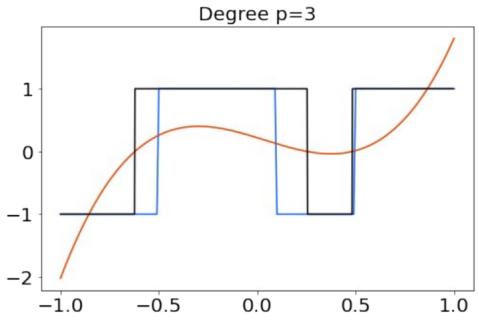
In order to minimize the least-squares equation, we need $Y = X^T\beta$, or $(X^T)^{-1}Y = \beta$. $(X^T)^\dagger Y = X(X^TX)^{-1}Y = XX^{-1}(X^T)^{-1}Y = (X^T)^{-1}Y$, so therefore, the two are equivalent.

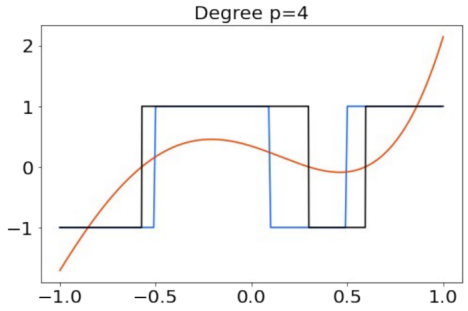
8 Problem 8

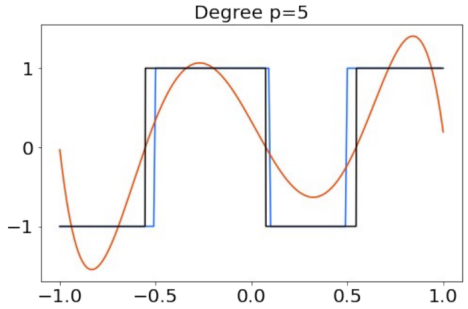
8.1 a and b

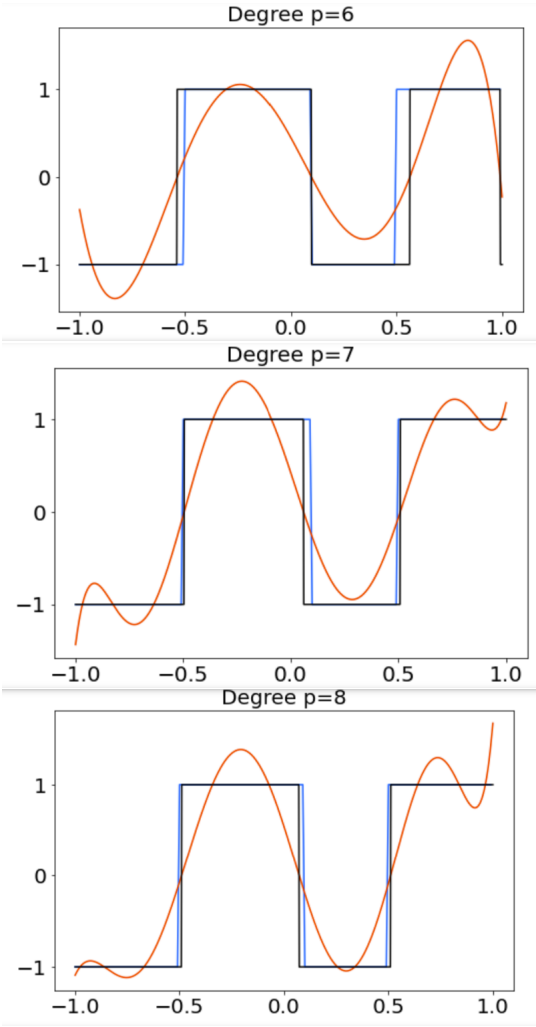



Degree p=2
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0


Degree p=3
1
0
−1
−2
−1.0
−0.5
0.0
0.5
1.0


Degree p=4
2
1
0
−1
−1.0
−0.5
0.0
0.5
1.0


Degree p=5
1
0
−1
−1.0
−0.5
0.0
0.5
1.0



8.2 c

The error rate decreases as the degree increases. This can be seen from the sign of the function matching the original function more and more as the degree increases.

The errors are:
Degree 0, Error: 0.0
Degree 1, Error: 55.00000000000001
Degree 2, Error: 55.00000000000001
Degree 3, Error: 49.5
Degree 4, Error: 55.00000000000001
Degree 5, Error: 55.00000000000001

Degree 6, Error: 55.00000000000001
Degree 7, Error: 55.00000000000001
Degree 8, Error: 55.00000000000001

```
import numpy as np
import matplotlib.pyplot as plt
import scipy
from scipy import io
from scipy import linalg
from scipy import optimize
from numpy.polynomial.polynomial import polyvander
from numpy.polynomial.polynomial import polyval
import seaborn as sns

fs=24
lw=4

rms = lambda x: np.sqrt(np.mean(np.square(x)))
```

Problem 4

Generate a random 20 × 10 matrix A and vector b ∈ R20, and compute x-hat = A†b.

Run the Richardson algorithm with μ = 1/‖A‖^2 for 500 iterations, and plot ‖x(k) − xˆ‖^2 to verify that x(k) is converging to x-hat.

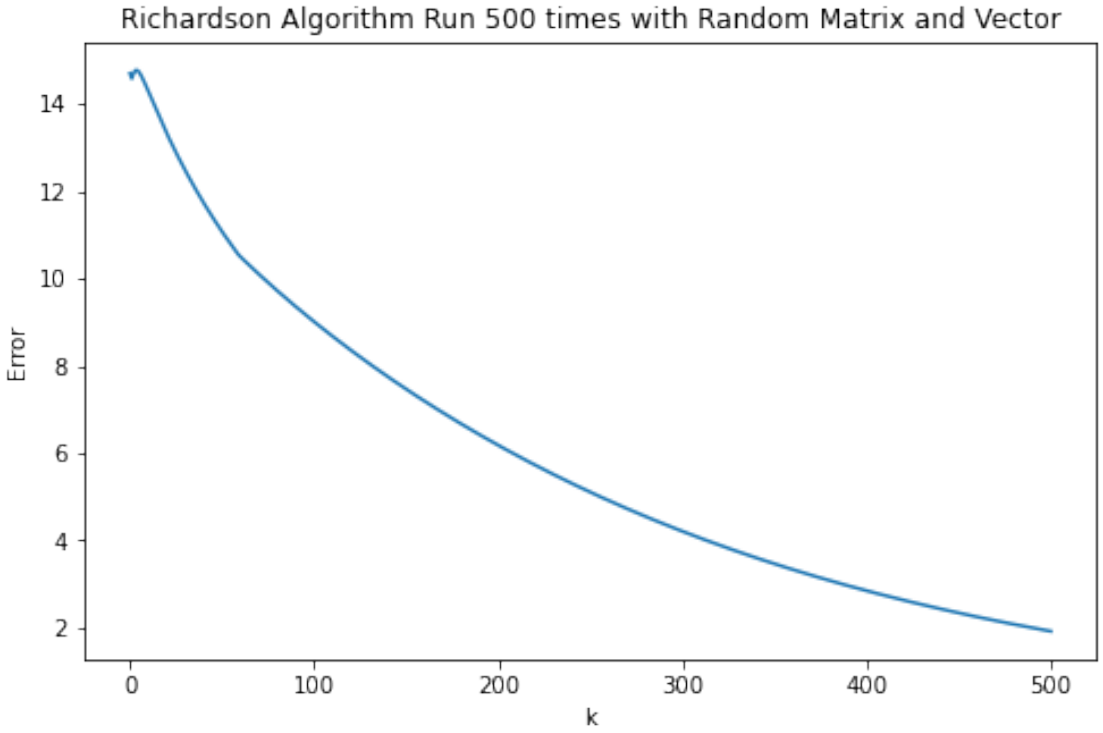
Just include your error plots. Plot on both a linear and log-scale.

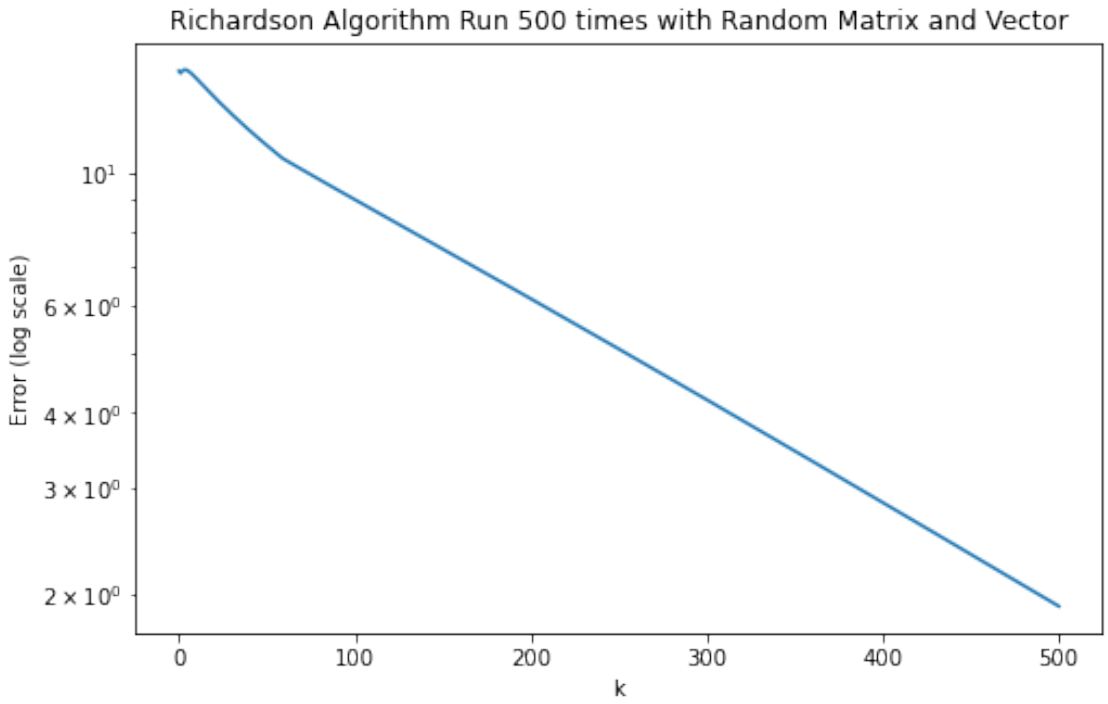
To get the log-scale you can use plt.yscale('log'). Include a snap shot of your code in your homework pdf. You can print your python notebook to a pdf.

```
A = np.random.rand(20, 10)
b = np.random.rand(20, 1)
#get expected x-hat val to compare
xhat =
np.matmul(np.matmul(np.linalg.inv(np.matmul(np.matrix.transpose(A),
A)), np.matrix.transpose(A)), b)

#initial val is 0 arr
xvals = []
x1 = np.zeros(10).reshape(10, 1)
xvals.append(x1)
index = 1
mu = 1/((np.linalg.norm(A, 1))**2)
#run the algorithm iter imes
iter = 500
for i in range(iter):
    prev = xvals[i]
    at = np.matrix.transpose(A)
    ax = np.matmul(A, prev)
    axb = np.subtract(ax, b)
    at = at*mu
    axbt = np.matmul(at, axb)
```

```
    res = np.subtract(xvals[i], axbt)
    xvals.append(res)

difs = []
for i in range(len(xvals)):
    dif = np.subtract(xvals[i], xhat)
    norm = (np.linalg.norm(dif, 1))**2
    difs.append(norm)

plt.figure(figsize=(8,5))
plt.plot(range(len(difs)), difs)
plt.ylabel("Error")
plt.xlabel("k")
plt.title("Richardson Algorithm Run " + str(iter) + " times with
Random Matrix and Vector")

plt.figure(figsize=(8,5))
plt.yscale('log')
plt.plot(range(len(difs)), difs)
plt.ylabel("Error (log scale)")
plt.xlabel("k")
plt.title("Richardson Algorithm Run " + str(iter) + " times with
Random Matrix and Vector")
```

```
Text(0.5, 1.0, 'Richardson Algorithm Run 500 times with Random Matrix
and Vector')
```

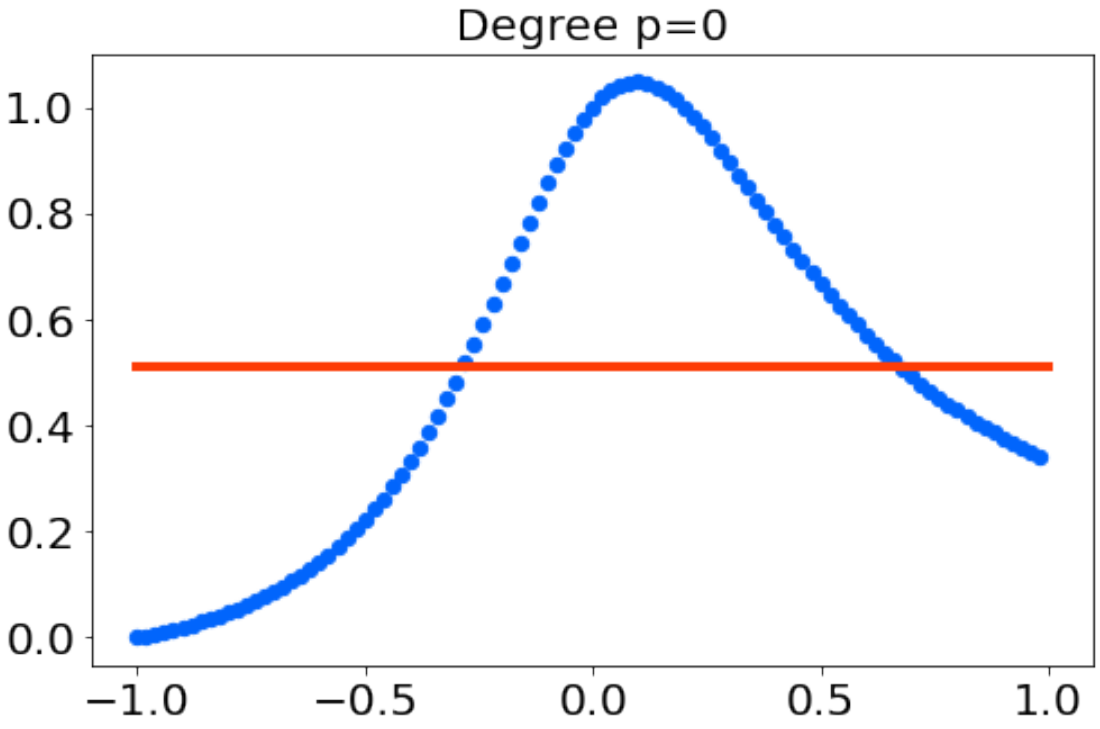


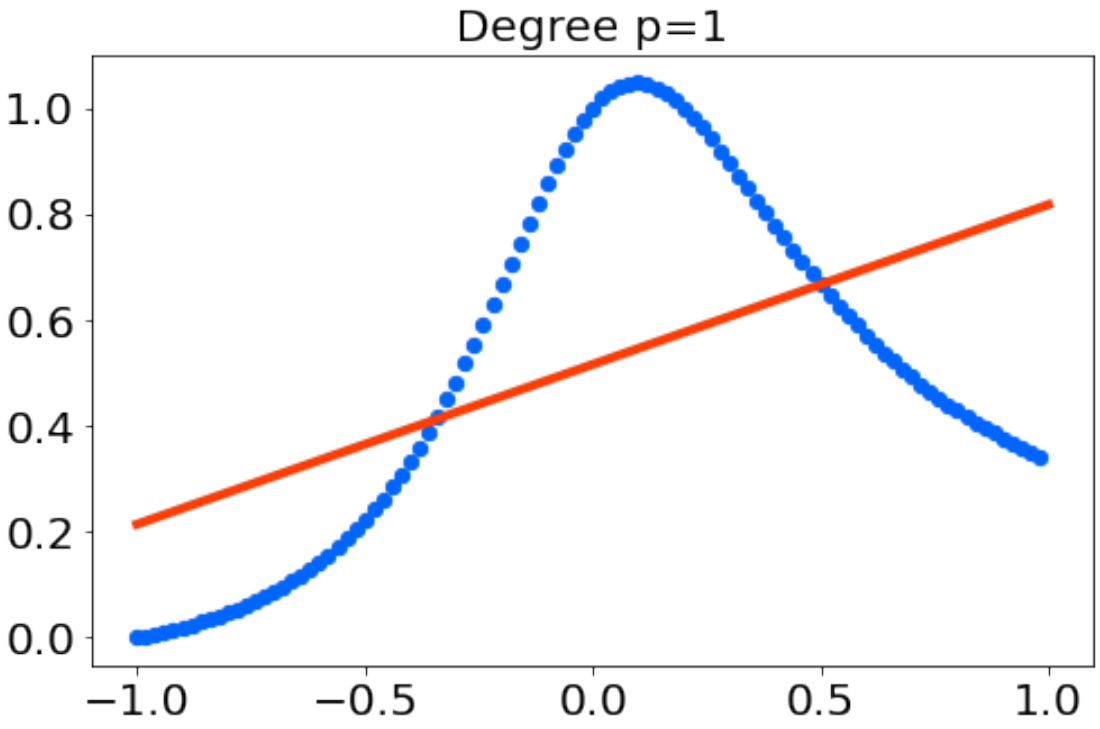


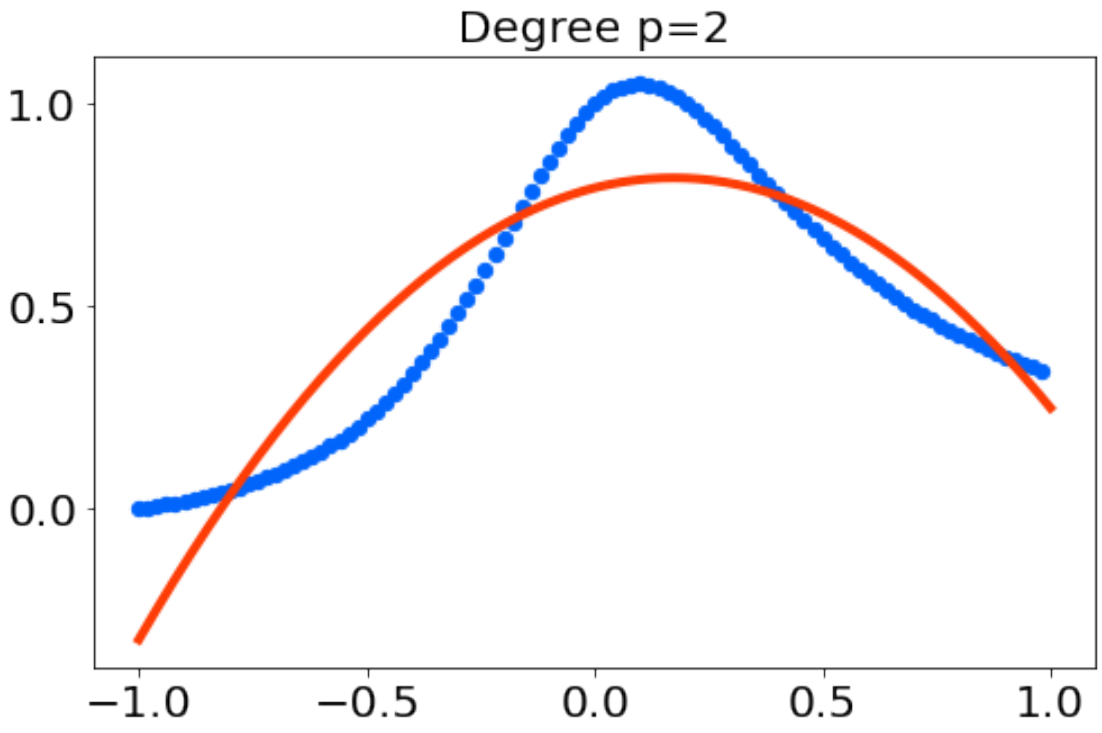
Problem 5

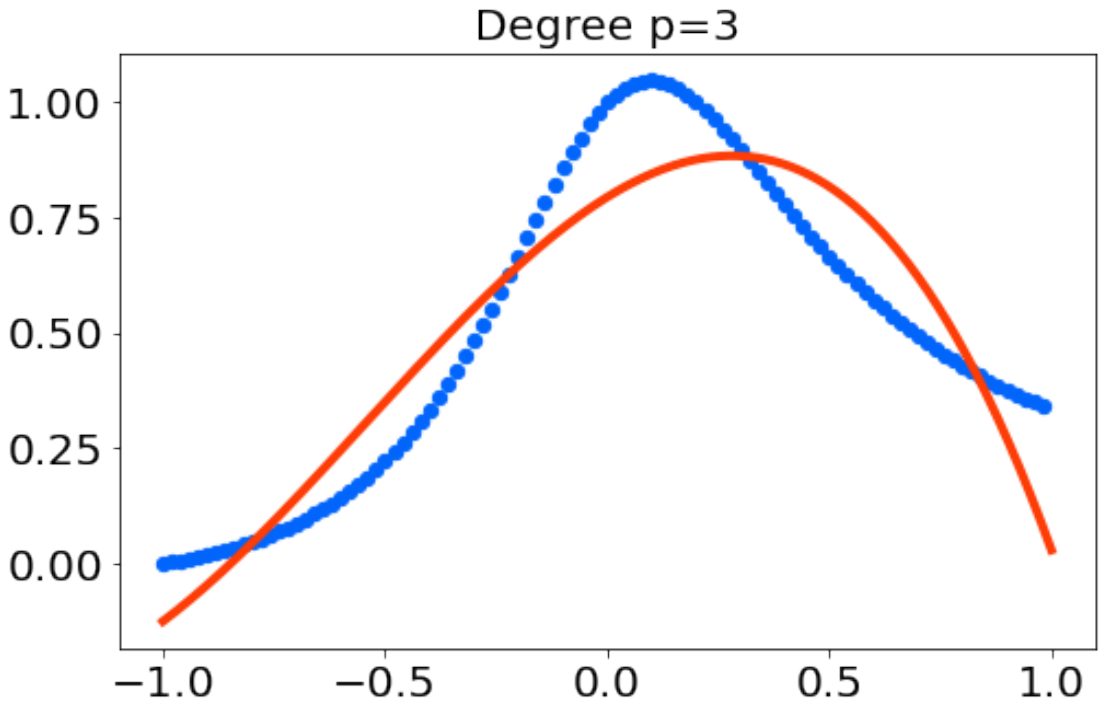
```
#Part A
#Setting up polynomial
np.random.seed(19)
eps=np.random.rand(100)
m = 100
i = 100
x = np.arange(-1, 1, 2/i)
y = np.zeros(i)
for i in range(100):
    y[i] = (1+x[i])/(1+5*x[i]**2)

#plotting different degrees
theta = []
eval_theta = []
As = []
tlin=np.linspace(-1,1, 1000)
for p in range(9):
    As.append(polyvander(x.flatten(),p))
    theta.append(np.linalg.inv(As[-1].T@As[-1])@As[-1].T@y)
    eval_theta.append(As[-1]@theta[-1])
    plt.figure(figsize=(8,5))
    plt.scatter(x,y, s=40, color='xkcd:bright blue')
    A=polyvander(tlin.flatten(),p)
    plt.plot(tlin, A@theta[-1], linewidth=lw, color='xkcd:red orange')
```

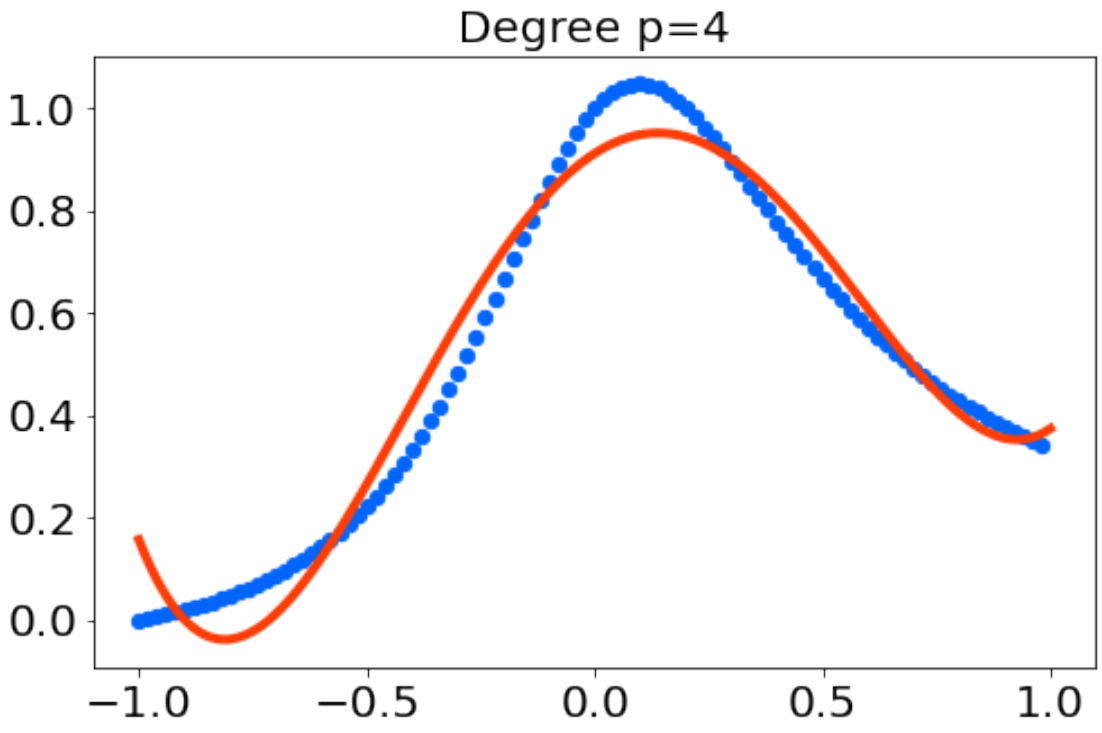
```
plt.tick_params(labelsize=fs-4)
plt.title('Degree p={}'.format(p), fontsize=fs-4)
```

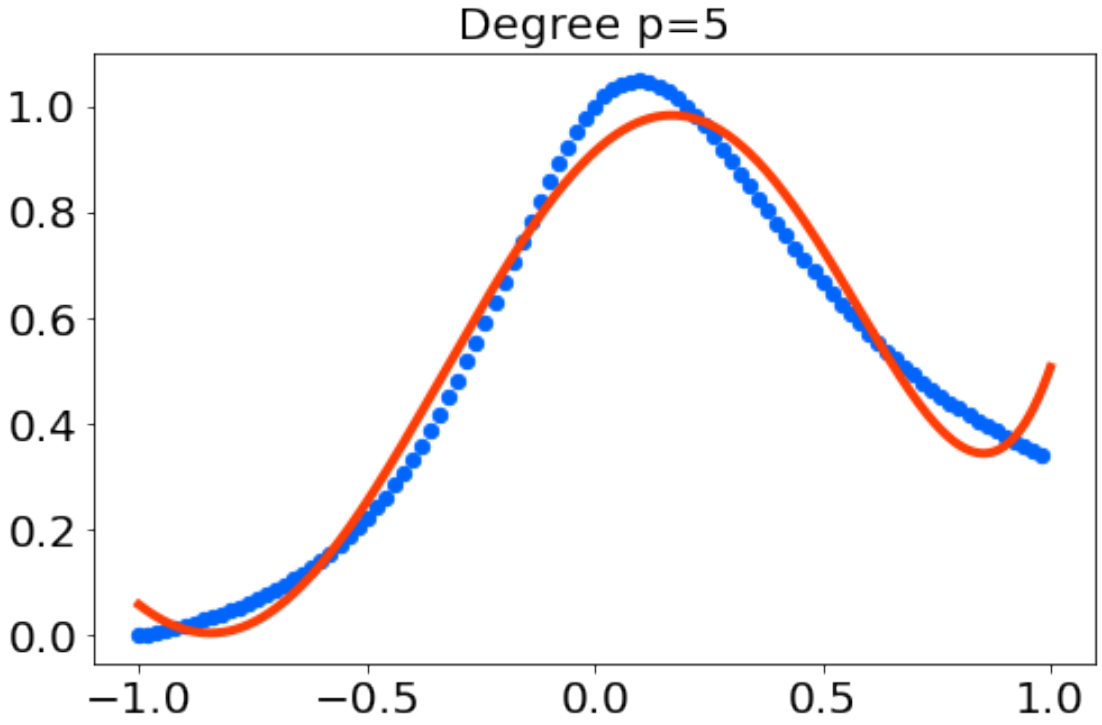


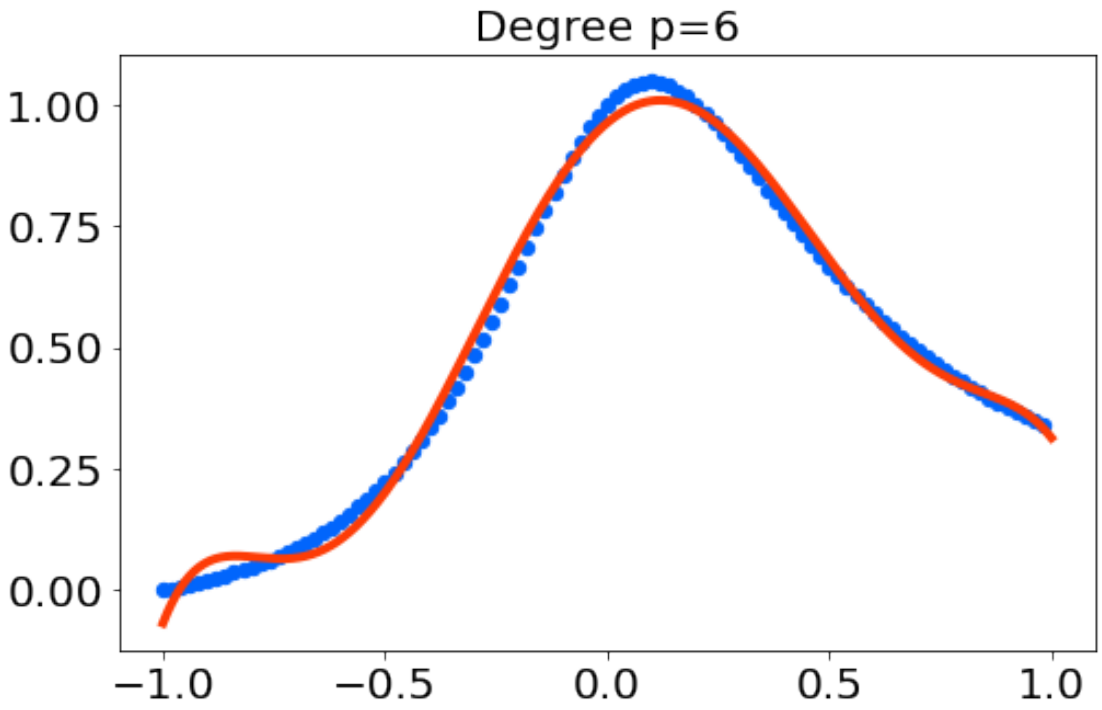


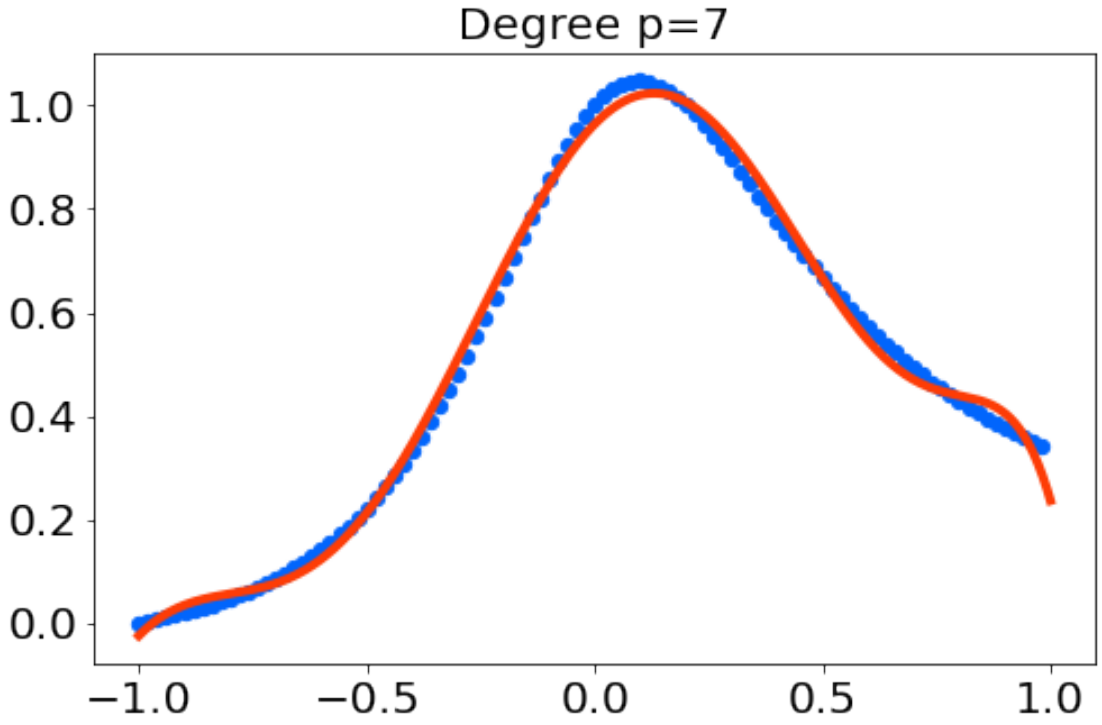

Degree p=2
1.0
0.5
0.0
−1.0
−0.5
0.0
0.5
1.0

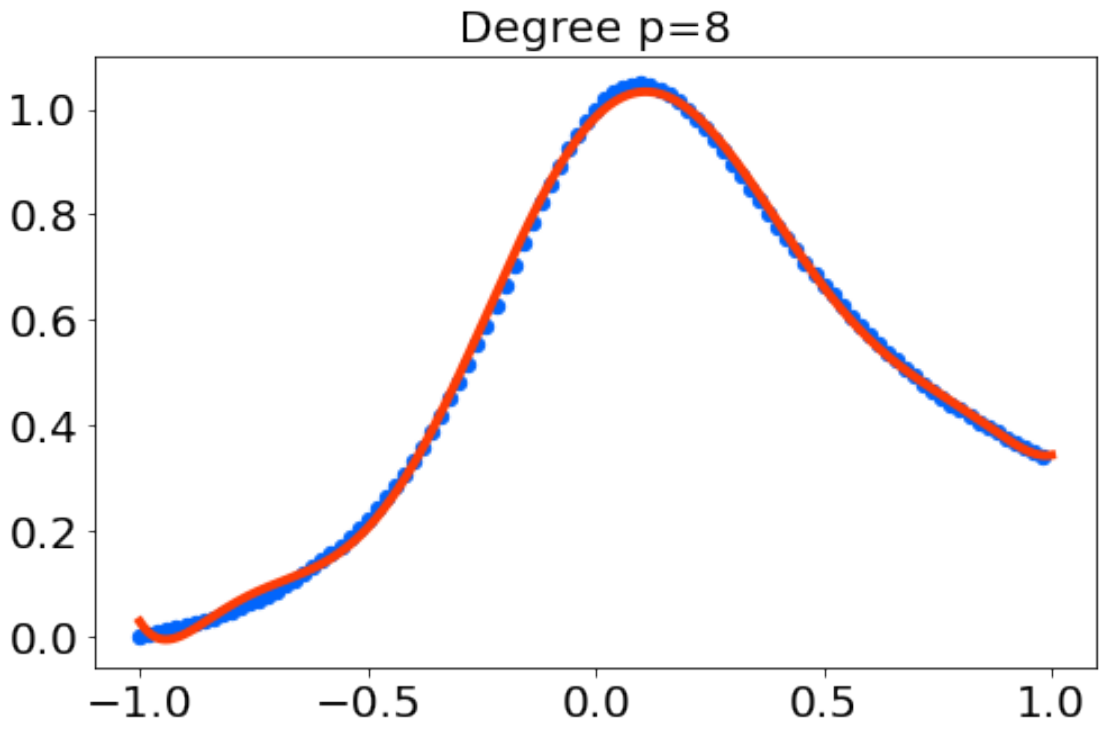

Degree p=3
1.00
0.75
0.50
0.25
0.00
−1.0
−0.5
0.0
0.5
1.0


Degree p=4
1.0
0.8
0.6
0.4
0.2
0.0
−1.0
−0.5
0.0
0.5
1.0


Degree p=5
1.0
0.8
0.6
0.4
0.2
0.0
−1.0
−0.5
0.0
0.5
1.0


Degree p=6
1.00
0.75
0.50
0.25
0.00
−1.0
−0.5
0.0
0.5
1.0


Degree p=7
1.0
0.8
0.6
0.4
0.2
0.0
−1.0
−0.5
0.0
0.5
1.0

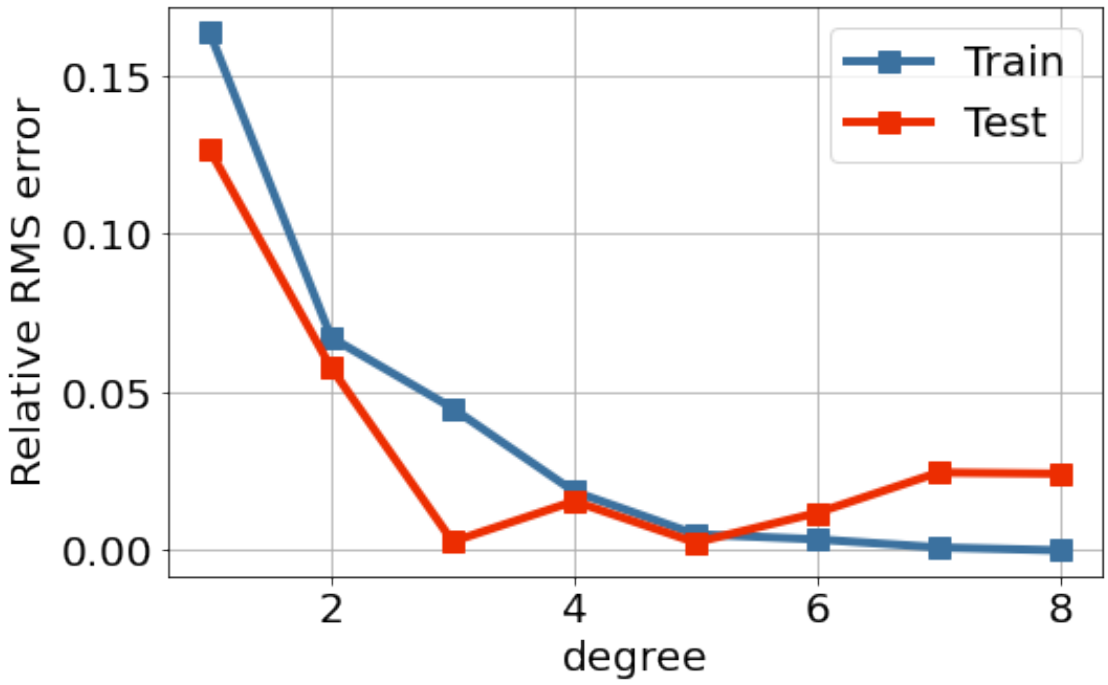



```
#Part B
u = np.arange(-1.1, 1.1, 2.2/10)
v = np.zeros(10)
for i in range(len(v)):
    v[i] = (1+u[i])/(1+5*u[i]**2)

m_train = 100
MAXORDER=8
t=-1+2*np.random.rand(m_train,1)
y_train=(1+t)/(1+5*t**2)*np.random.rand(m_train,1)
m_test = 10
t_test = -1 + 2*np.random.rand(m_test,1)
y_test = (1+t_test)/(1+5*t_test**2) + 0.025*np.random.randn(m_test,1)

error_train = np.zeros(MAXORDER)
error_test = np.zeros(MAXORDER)
for p in range(0,MAXORDER):
    A = polyvander(t.flatten(),p+1)
    theta = np.linalg.inv(A.T@A)@A.T@y_train
    error_train[p] = np.linalg.norm(A@theta - y_train) /
np.linalg.norm(y_train)
    error_test[p] = np.linalg.norm( polyvander(t_test.flatten(), p+1)
@ theta - y_test) / np.linalg.norm(y_test)

amin = np.amin(error_train)
error_train = error_train-amin
error_test = error_test-amin
```

```
plt.figure(figsize=(8,5))
plt.grid(True)
plt.plot(range(1,MAXORDER+1), error_train, label = "Train", marker
='s', markersize=10,linewidth=lw, color='xkcd:muted blue')
plt.plot(range(1,MAXORDER+1), error_test, label = "Test", marker ='s',
markersize=10,linewidth=lw, color='xkcd:tomato red')
plt.legend(fontsize=fs-4)
plt.tick_params(labelsize=fs-4)
plt.xlabel('degree', fontsize=fs-4)
plt.ylabel('Relative RMS error', fontsize=fs-4)
```

```
print("This tells us the reasonable degree fit is around 4-6, as the
least difference in degree between train and test occurs among those
values")
```

```
This tells us the reasonable degree fit is around 4-6, as the least
difference in degree between train and test occurs among those values
```



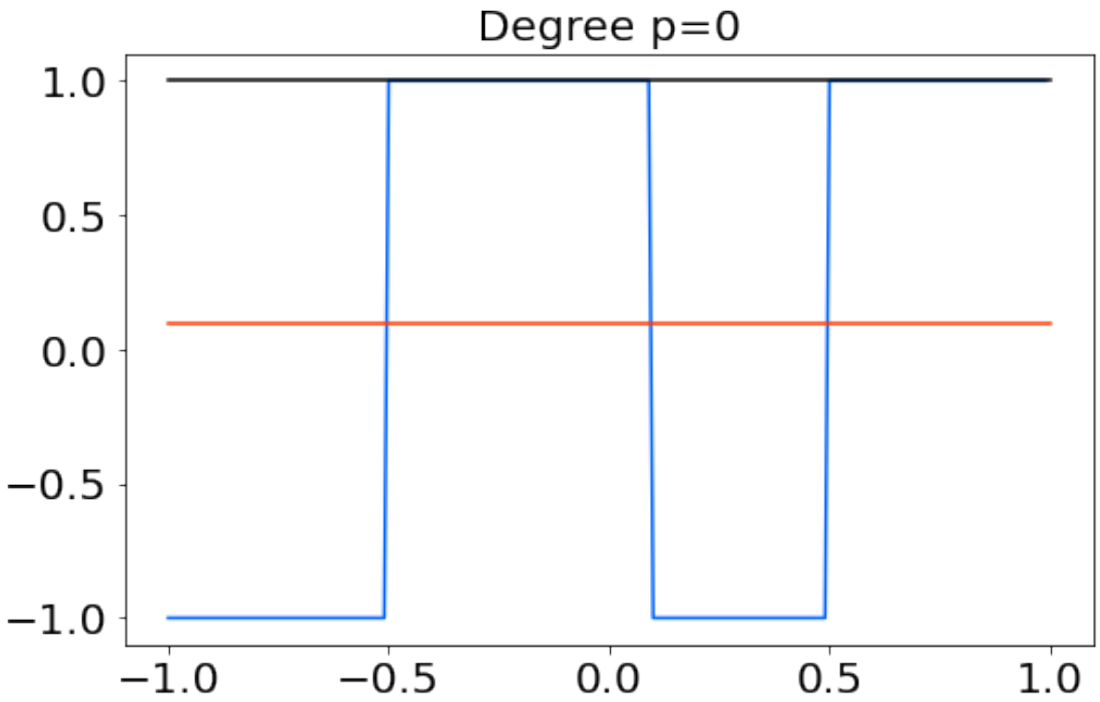
Problem 8

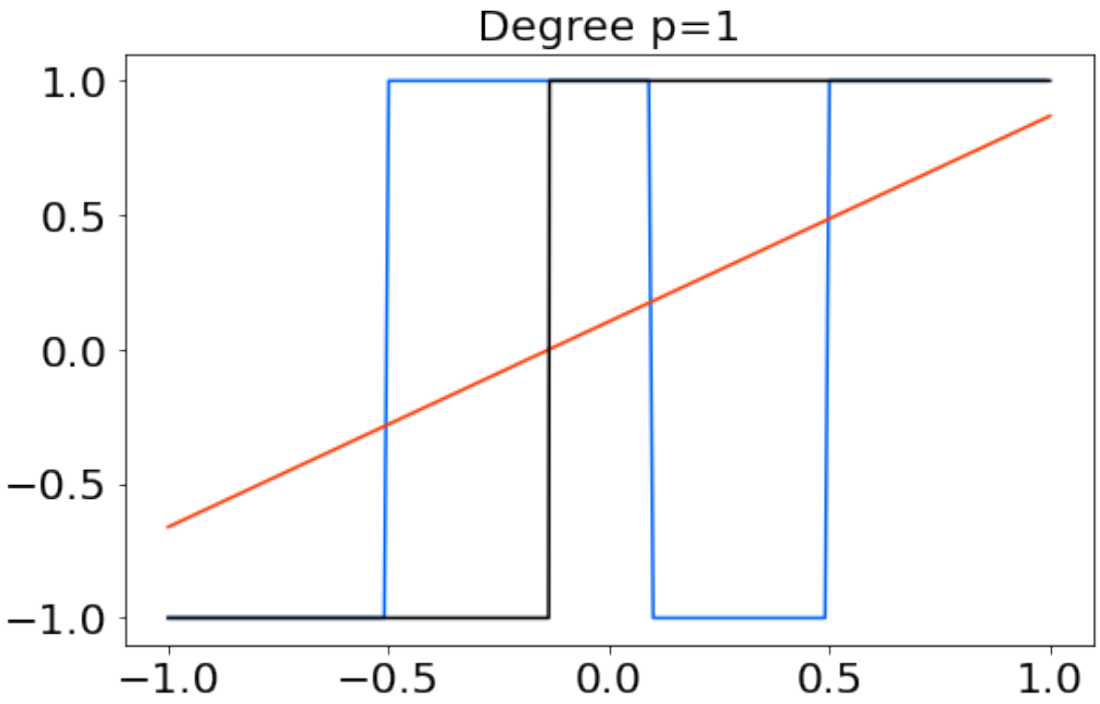
```
#Functions used later
numTP = lambda y,yhat: sum([1 for i in range(len(y)) if y[i] == True
and yhat[i] == True])
numFN = lambda y,yhat: sum([1 for i in range(len(y)) if y[i] == True
and yhat[i] == False])
numFP = lambda y,yhat: sum([1 for i in range(len(y)) if y[i] == False
and yhat[i] == True])
```

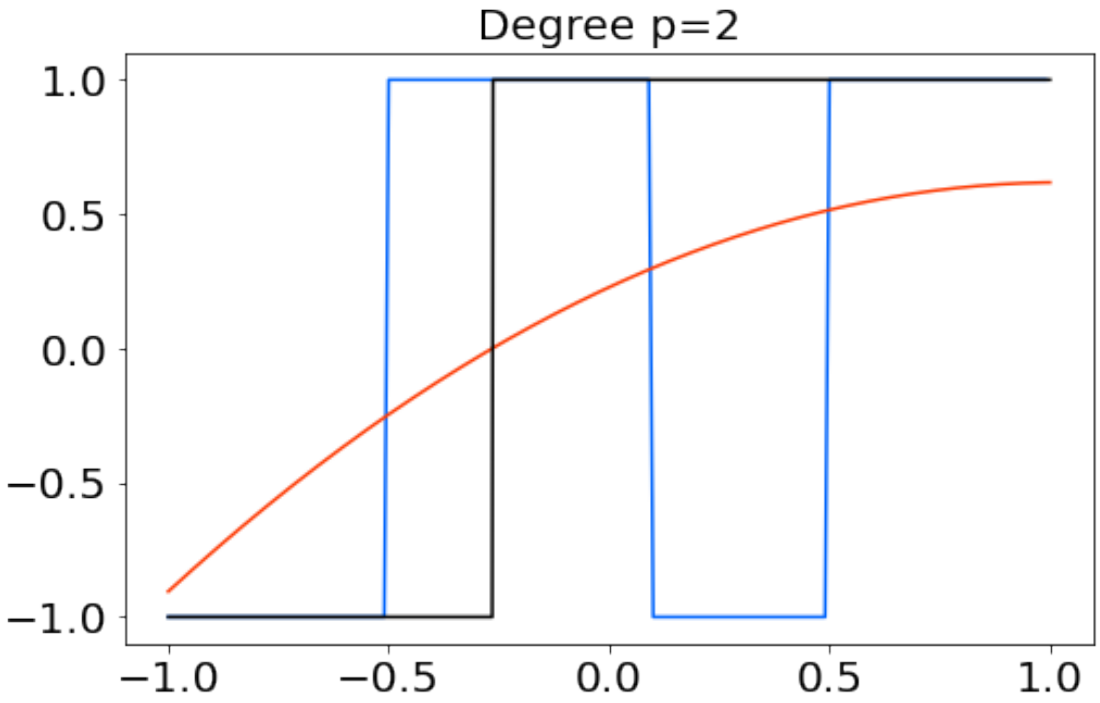
```
numTN = lambda y,yhat: sum([1 for i in range(len(y)) if y[i] == False
and yhat[i] == False])
confusion_matrix = lambda y,yhat:
np.vstack([[numTP(y,yhat),numFN(y,yhat)],
[numFP(y,yhat),numTN(y,yhat)]])
error_rate = lambda y,yhat: (numFN(y,yhat) + numFP(y,yhat)) / len(y)
error_rate2 = lambda y,yhat: np.average(y != yhat)


#Part A
#Setting up polynomial
np.random.seed(19)
eps=np.random.rand(100)
m = 100
j = 200
x = np.arange(-1, 1, 2/j)
y = np.zeros(j)
for i in range(200):
    if (-0.5 <= x[i] and x[i] < 0.1) or (0.5 <= x[i]):
        y[i] = 1
    else:
        y[i] = -1

#plotting different degrees
theta = []
eval_theta = []
As = []
tlin=np.linspace(-1,1, 1000)
for p in range(9):
    As.append(polyvander(x.flatten(),p))
    currthet = np.linalg.inv(As[-1].T@As[-1])@As[-1].T@y
    theta.append(currthet)
    curras = As[-1]@theta[-1]
    eval_theta.append(curras)
    plt.figure(figsize=(8,5))
    plt.plot(x,y, color='xkcd:bright blue')
    A=polyvander(tlin.flatten(),p)
    Asign = np.sign(A@theta[-1])
    plt.plot(tlin, A@theta[-1], color='xkcd:red orange')
    plt.plot(tlin, Asign, color='xkcd:black')
    plt.tick_params(labelsize=fs-4)
    plt.title('Degree p={}'.format(p), fontsize=fs-4)

    yhat = np.matmul(A, currthet) > 0
    #Part C- Errors
    error = error_rate(y,yhat)*100
    print('Degree {}'.format(p) + ", Error: " + str(error))
```

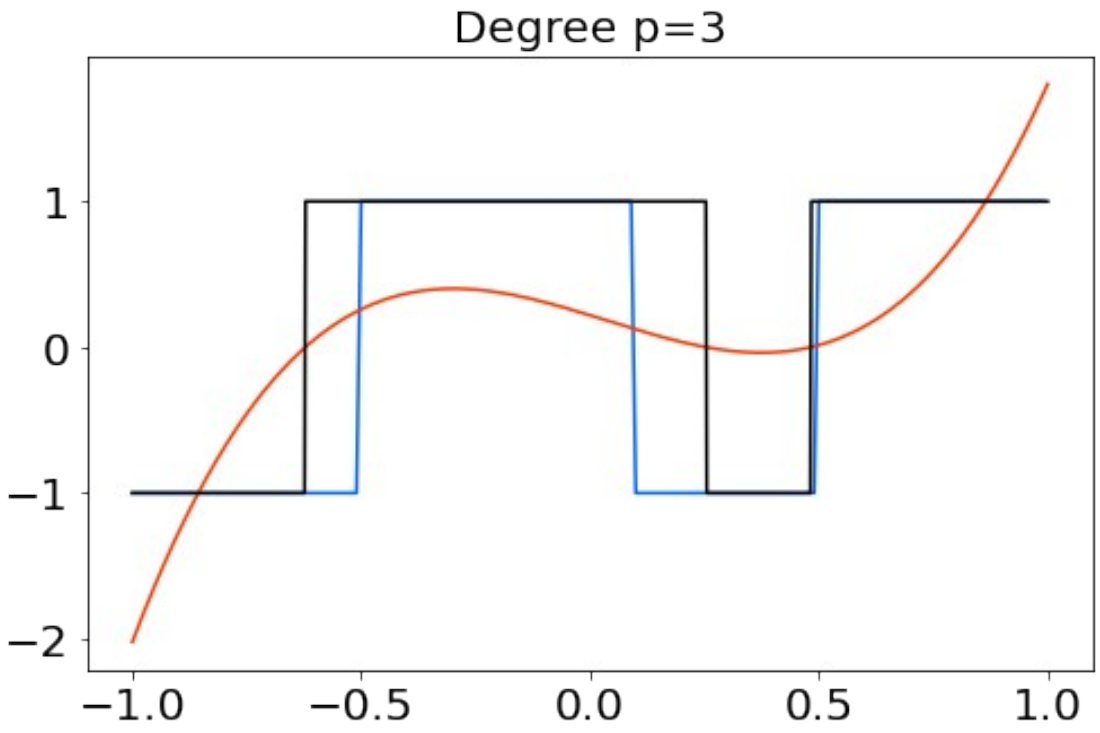
```
Degree 0, Error: 0.0
Degree 1, Error: 55.00000000000001
```

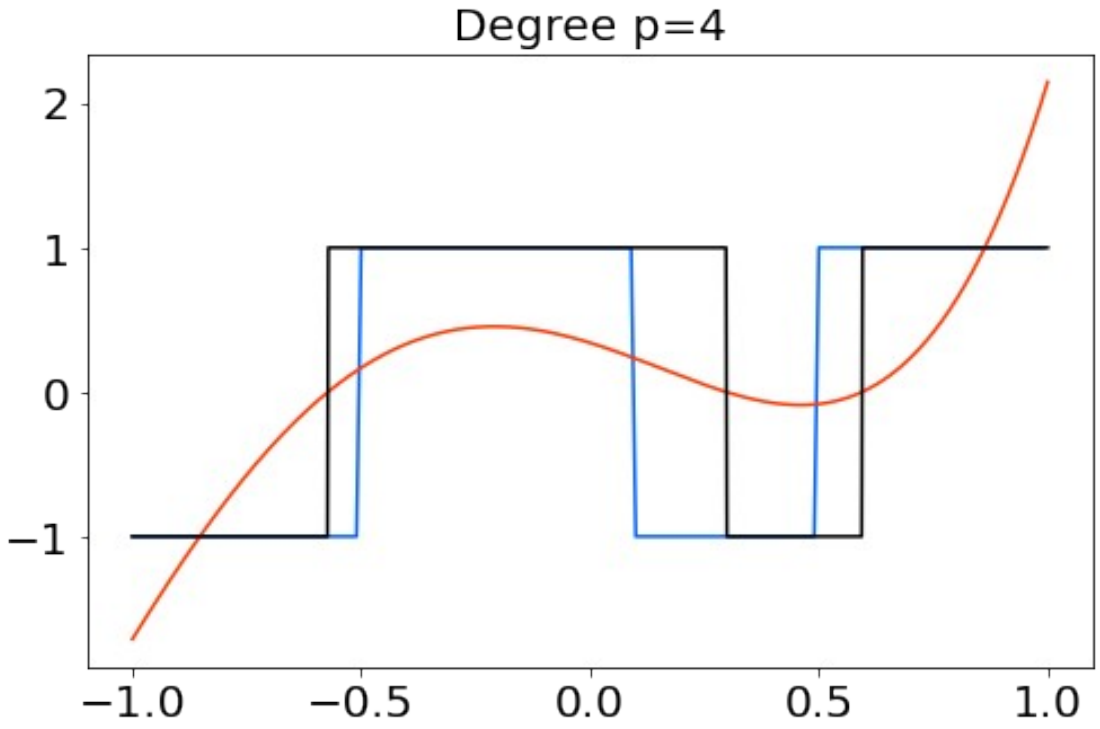
```
Degree 2, Error: 55.00000000000001
Degree 3, Error: 49.5
Degree 4, Error: 55.00000000000001
Degree 5, Error: 55.00000000000001
Degree 6, Error: 55.00000000000001
Degree 7, Error: 55.00000000000001
Degree 8, Error: 55.00000000000001
```

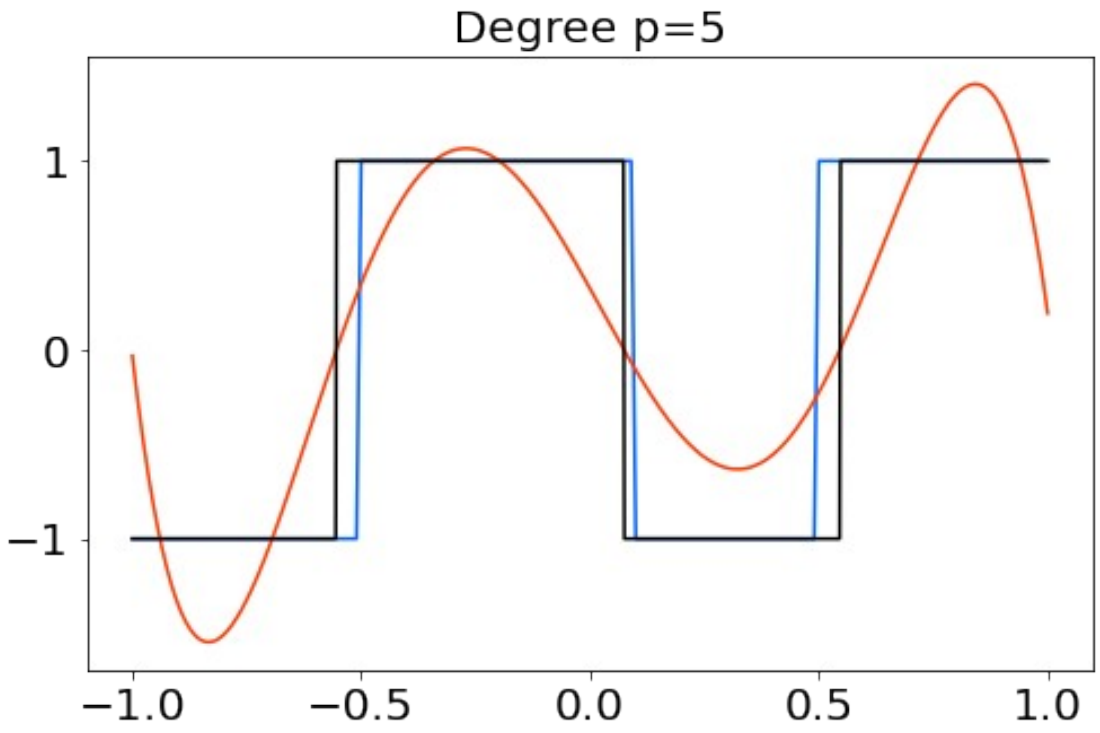


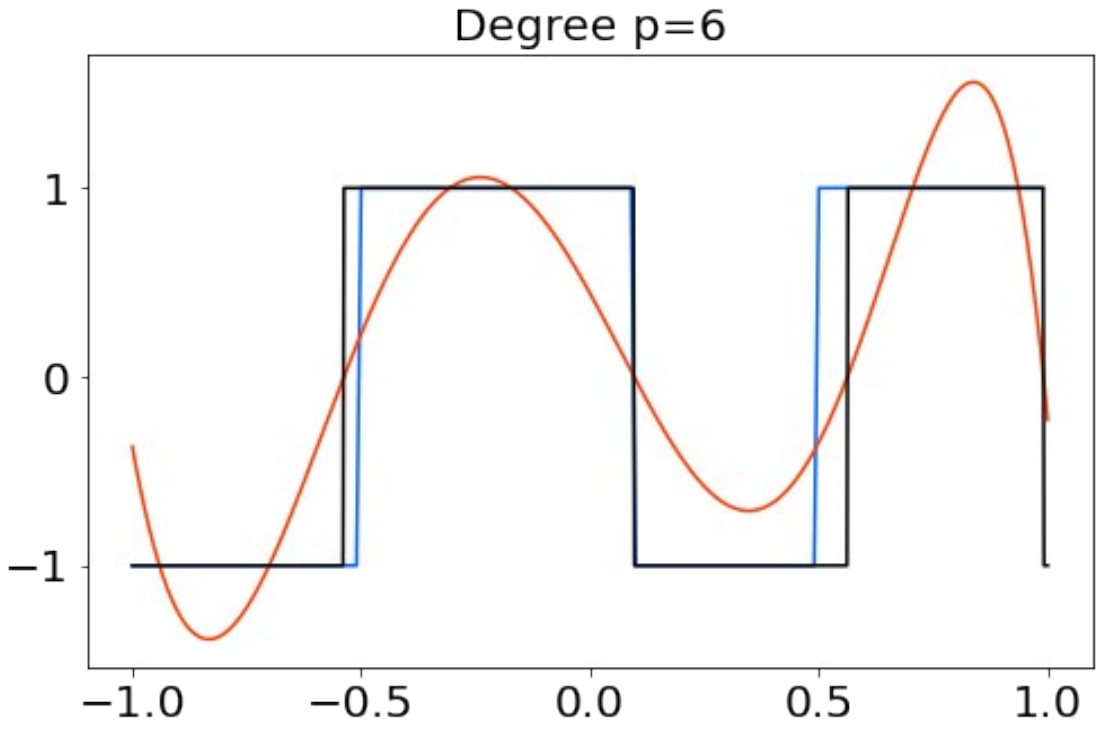

Degree p=1
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0

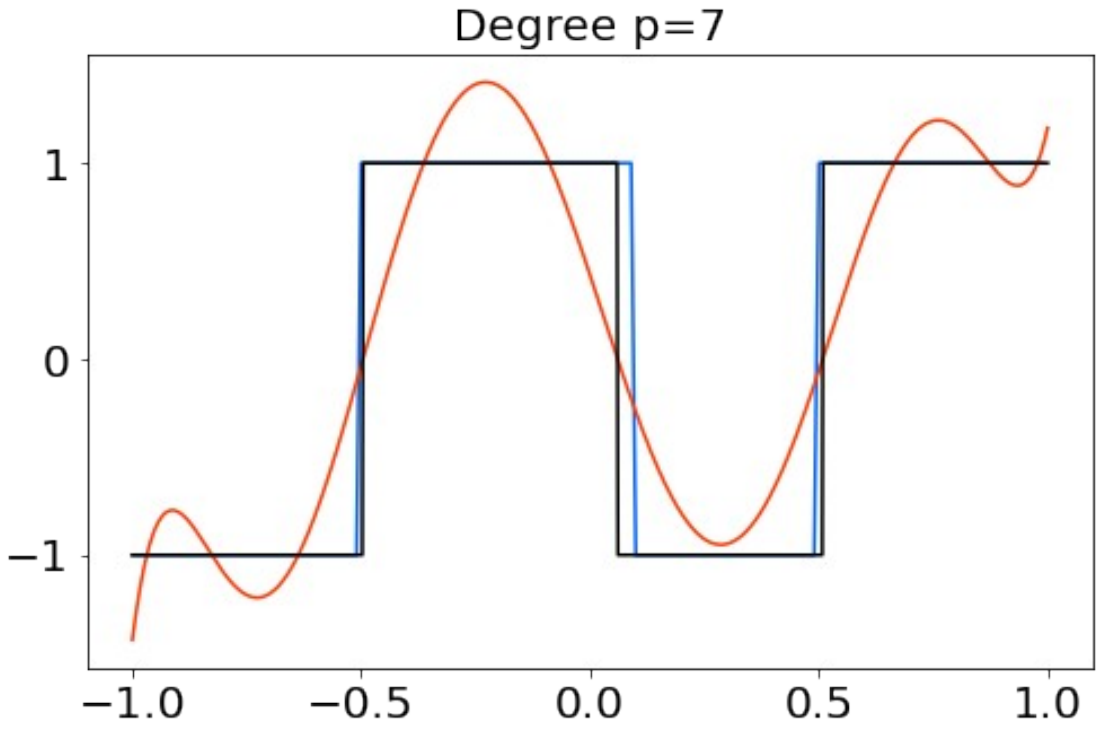

Degree p=2
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0

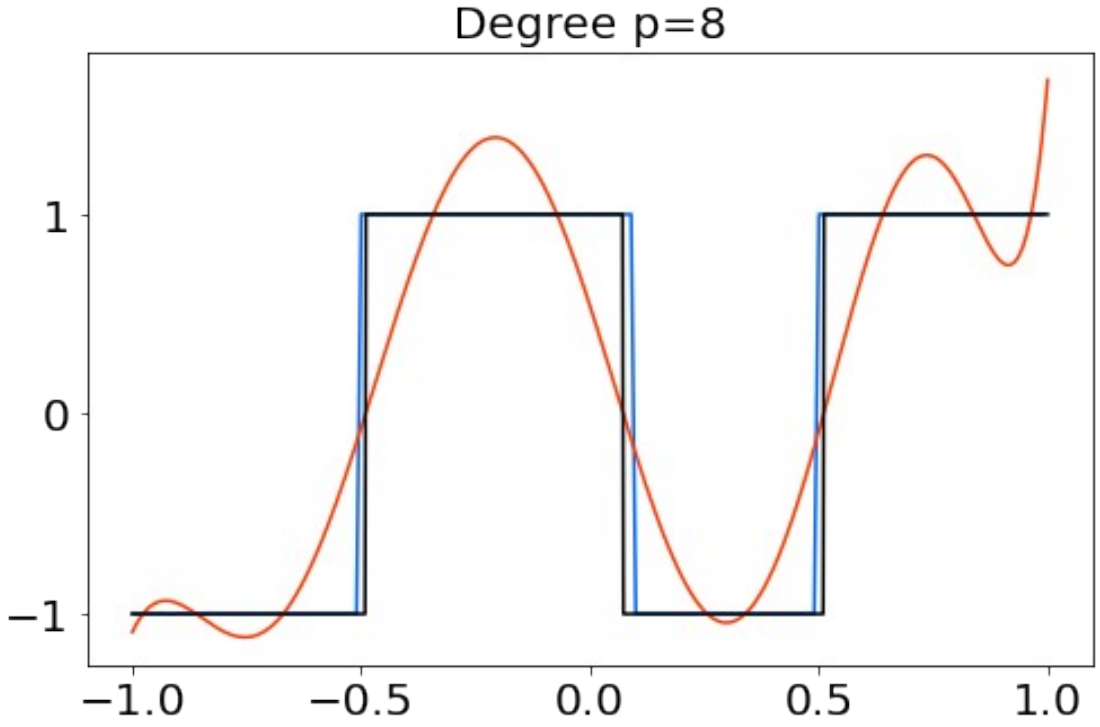

Degree p=3
1
0
−1
−2
−1.0
−0.5
0.0
0.5
1.0


Degree p=4
2
1
0
−1
−1.0
−0.5
0.0
0.5
1.0


Degree p=5
1
0
−1
−1.0
−0.5
0.0
0.5
1.0


Degree p=6
1
0
−1
−1.0
−0.5
0.0
0.5
1.0


Degree p=7
1
0
−1
−1.0
−0.5
0.0
0.5
1.0


Degree p=8
1
0
−1
−1.0
−0.5
0.0
0.5
1.0