In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.linalg
```

In [ ]:
```
K = 3

theta_true = np.ones((K,))
a = 0
b = 1
N = 100

X = np.linspace(a, b, N)
```

In [ ]:
```
def phi(X, K):
    acc = len(X)
    V = np.zeros((acc, K))
    for i in range(K):
        V[:, i] = X**i
    return V
```

In [ ]:
```
var = 0.1
e = np.random.normal(0, var, N)
Y = phi(X, K) @ theta_true + e
```

In [ ]:
```
D = (X, Y)
```

In [ ]:
```
def GD(x0, D, grad_f, tolf=1e-6, tolx=1e-6, kmax=10000000, alpha=1e-5, lam=1):
    k = 0
    (X, Y) = D
    xk = x0

    xs = [xk]
    grad_vals = [grad_f(X, Y, xk, lam)]

    while (np.linalg.norm(grad_f(X, Y, xk, lam)) >= tolf and k < kmax - 1):
        x_prec = xk
        xk = x_prec - (alpha * grad_f(X, Y, x_prec, lam))

        xs.append(xk)
        grad_vals.append(grad_f(X, Y, xk, lam))

        if np.linalg.norm(grad_vals[-1]) < tolf:
            break

        if np.linalg.norm(xk - xs[-1]) < tolx:
            break

        k+=1

    return xk

def shuffle(X, Y):
    N = len(X)
    idx = np.arange(N)
    np.random.shuffle(idx)

    new_X = X[idx]
```

```
    new_Y = Y[idx]

    return new_X, new_Y

def SGD(w0, D, grad_f, batch_size = 5, n_epochs = 10, alpha = 1e-5, lam=1):
    (X, Y) = D
    N = len(X)

    n_batch_per_epoch = N // batch_size

    w = np.array(w0)
    ws = [w]

    for epoch in range(n_epochs):
        X_new, Y_new = shuffle(X, Y)

        for batch in range(n_batch_per_epoch):
            n = batch * batch_size
            m = (batch + 1) * batch_size
            Mx = X_new[n : m]
            My = Y_new[n : m]

            w = w - (alpha * grad_f(Mx, My, w, lam))
            ws.append(w)

    return w

def grad_f_MLE(X, Y, theta, lam=None):
    K = len(theta)
    return phi(X, K).T @ ((phi(X, K) @ theta) - Y)

def theta_calc_eq_ne(phi_X, Y):
    first_fact = phi_X.T @ phi_X
    second_fact = phi_X.T @ Y
    try:
        L = scipy.linalg.cholesky(first_fact, lower = True)
        y = scipy.linalg.solve_triangular(L, b, lower = True)
        theta_found_normeq = scipy.linalg.solve_triangular(L.T, y)
    except:
        theta_found_normeq = np.linalg.solve(first_fact, second_fact)
    return theta_found_normeq
```

In [ ]:

```
mean = 1

def MLE(D, K, method):
    X, Y = D
    theta = None
    if method == "NE":
        theta = theta_calc_eq_ne(phi(X, K), Y)
    if method == "GD":
        w0 = np.random.normal(mean, var, K)
        theta = GD(w0, D, grad_f_MLE)
    if method == "SGD":
        w0 = np.random.normal(mean, var, K)
        theta = SGD(w0, D, grad_f_MLE)
    return theta
```

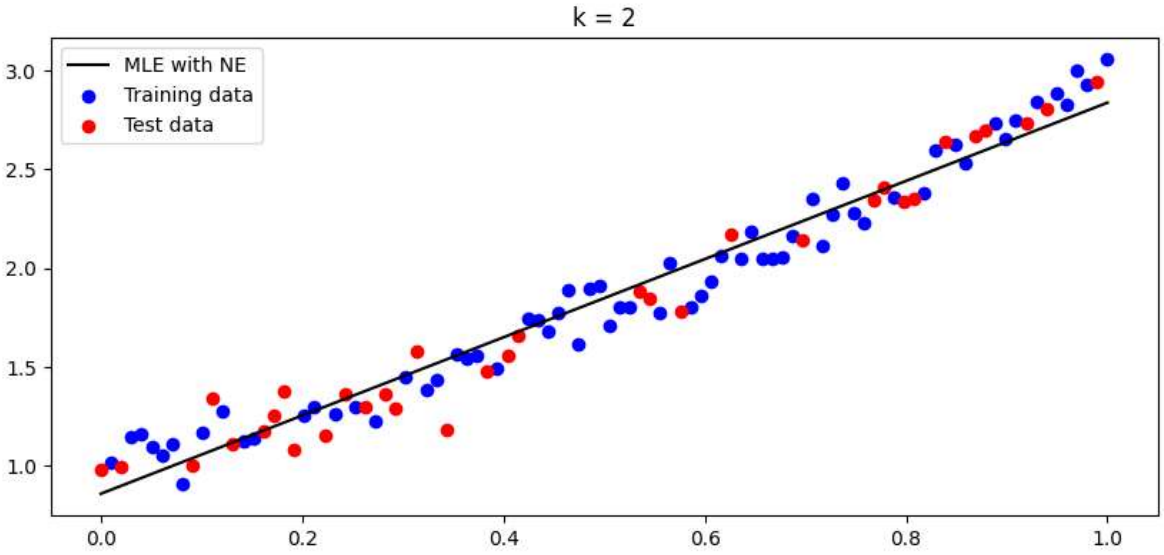
In [ ]:

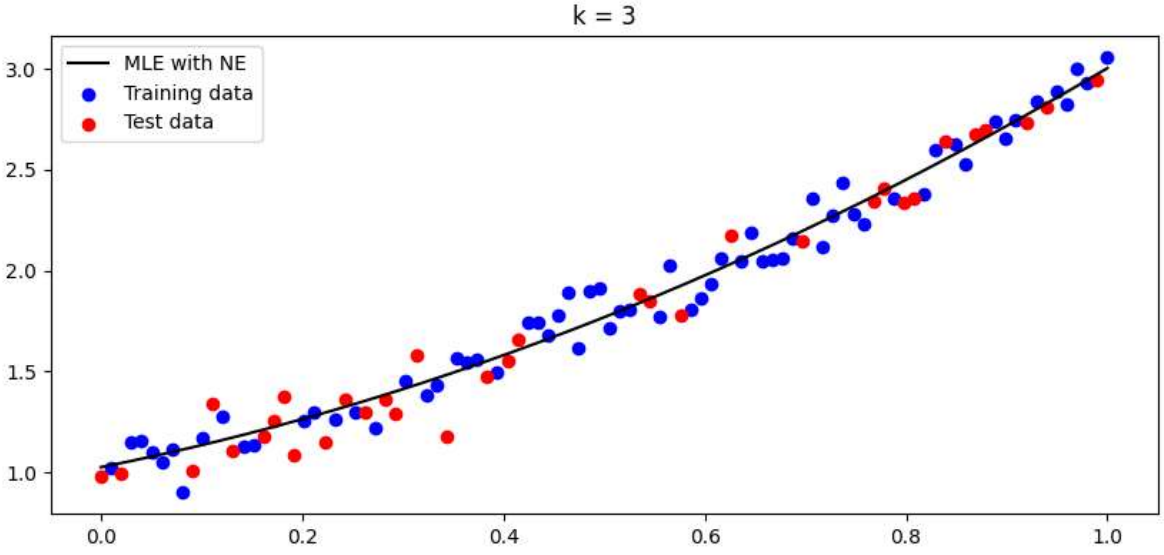
```
def avg_abs_err(theta, D):
    X, Y = D
```

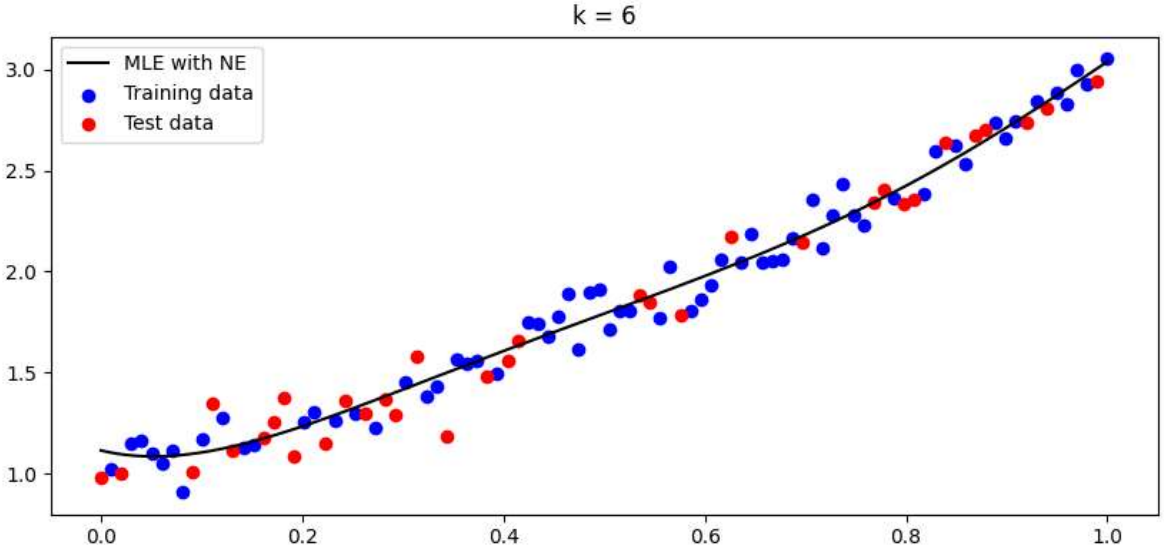
```
    K = len(theta)
    return (np.linalg.norm((phi(X, K) @ theta) - Y)**2) * (1/N)
```

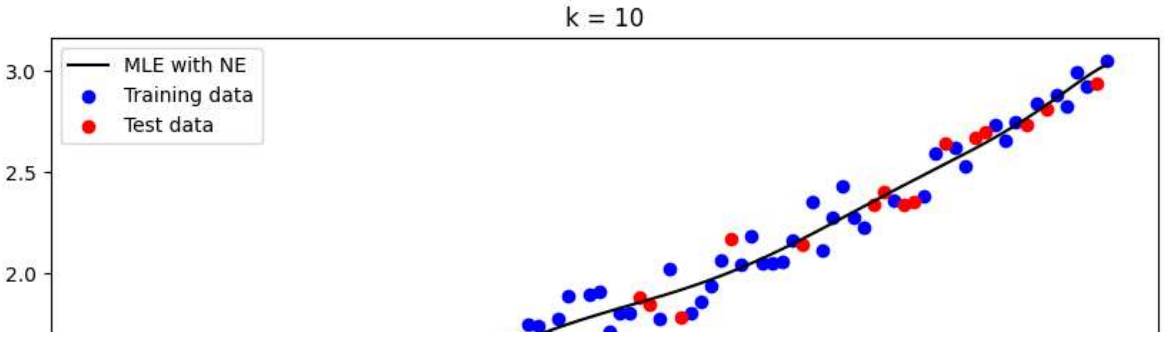
In [ ]:
```
N_train = int(N / 3 * 2)

def  train_test_split(X, Y, N_train):
    N = len(X)

    idx = np.arange(N)
    np.random.shuffle(idx)

    train_idx = idx[:N_train]
    test_idx = idx[N_train:]

    Xtrain = X[train_idx]
    Ytrain = Y[train_idx]

    Xtest = X[test_idx]
    Ytest = Y[test_idx]

    return Xtrain, Xtest, Ytrain, Ytest

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, N_train)
```

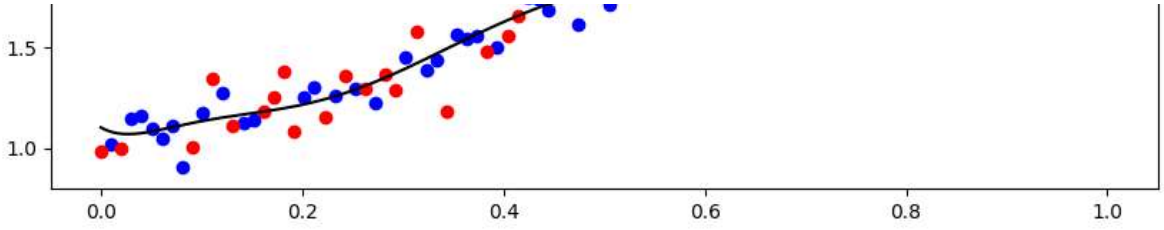
In [ ]:
```
ks_MLE = [2, 3, 6, 10]
thetas_MLE = []
methods = ['NE', 'NE', 'NE', 'NE']
errs_train_MLE = []
errs_test_MLE = []

for i, k in enumerate(ks_MLE):
    thetas_MLE.append(MLE((X_train, Y_train), k, methods[i]))
    errs_train_MLE.append(avg_abs_err(thetas_MLE[-1], (X_train, Y_train)))
    errs_test_MLE.append(avg_abs_err(thetas_MLE[-1], (X_test, Y_test)))

fig, ax = plt.subplots(len(ks_MLE), figsize=(10, 20))

for i in range(len(ks_MLE)):
    X_plot = np.linspace(a, b, 1000)
    Y_plot = phi(X_plot, ks_MLE[i]) @ thetas_MLE[i]
    ax[i].plot(X_plot, Y_plot, label=f'MLE with {methods[i]}', color='k')
    ax[i].scatter(X_train, Y_train, label='Training data', color='b')
    ax[i].scatter(X_test, Y_test, label='Test data', color='r')
    ax[i].set_title('k = ' + str(ks_MLE[i]))
    ax[i].legend()
```


k = 2
MLE with NE
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0


k = 3
MLE with NE
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0


k = 6
MLE with NE
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0


k = 10
MLE with NE
Training data
Test data
3.0
2.5
2.0

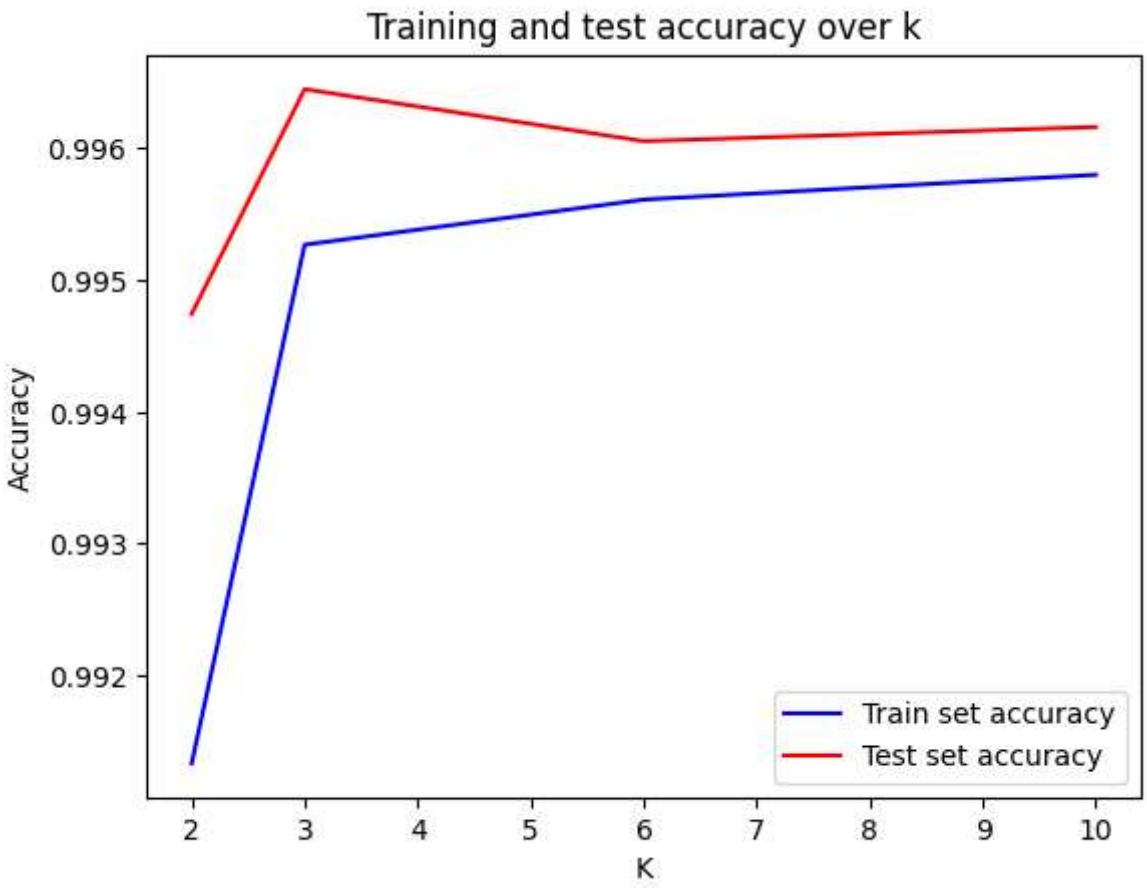


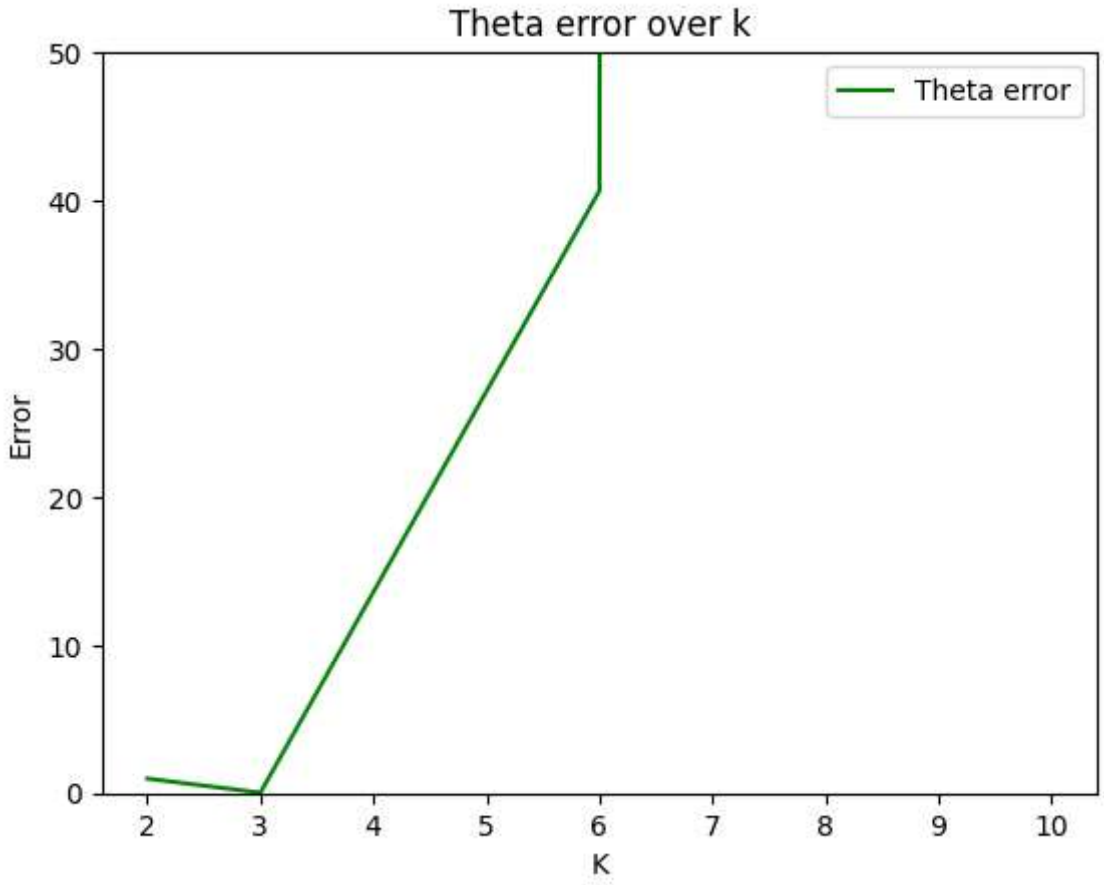
In [ ]:
```
def theta_error(theta):
    ktry = len(theta)
    theta_true = np.ones((ktry,))
    return np.linalg.norm(theta - theta_true, 2, axis=0)

thetaerror = [theta_error(theta) for theta in thetas_MLE]
print(thetaerror)
```

```
[0.987153282518485, 0.03279669138171278, 40.657354030791446, 30134.427605087247]
```

In [ ]:
```
plt.plot(ks_MLE, errs_train_MLE, label = "Train set error", color='b')
plt.plot(ks_MLE, errs_test_MLE, label = "Test set error", color='r')
plt.legend()
plt.title("Training and test error over k")
plt.xlabel("K")
plt.ylabel("Error")
plt.show()

acctrain = [1-e for e in errs_train_MLE]
acctest = [1-e for e in errs_test_MLE]
plt.plot(ks_MLE, acctrain, label = "Train set accuracy", color='b')
plt.plot(ks_MLE, acctest, label = "Test set accuracy", color='r')
plt.legend()
plt.title("Training and test accuracy over k")
plt.xlabel("K")
plt.ylabel("Accuracy")
plt.show()

plt.plot(ks_MLE, thetaerror, label = "Theta error", color='g')
plt.legend()
plt.title("Theta error over k")
plt.xlabel("K")
plt.ylabel("Error")
plt.ylim(0, 50)
plt.show()
```


Training and test error over k
Train set error
Test set error
0.008
0.007
0.006
0.005
0.004
Error
2
3
4
5
6
7
8
9
10
K

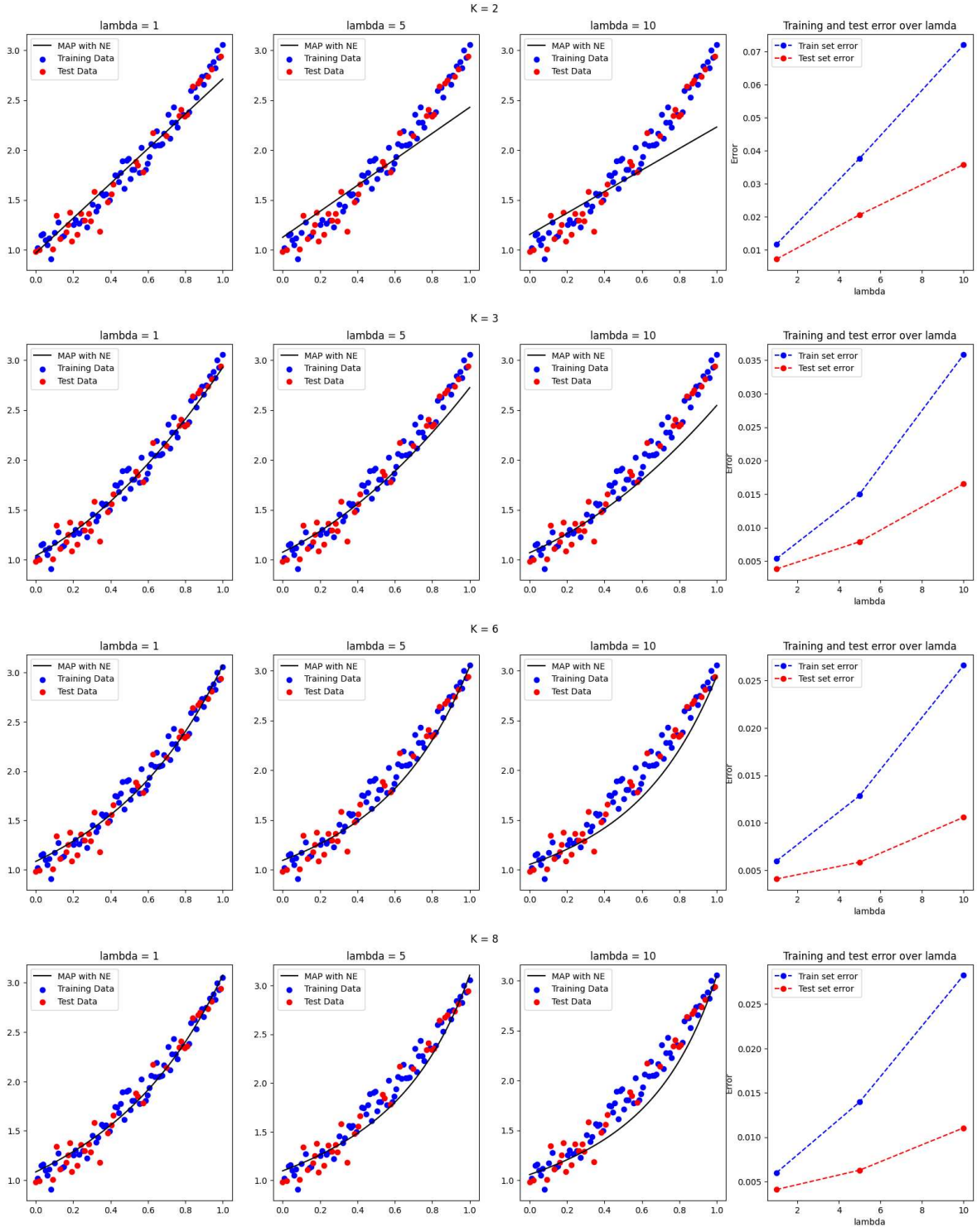

Training and test accuracy over k
0.996
0.995
0.994
0.993
0.992
Accuracy
Train set accuracy
Test set accuracy
2
3
4
5
6
7
8
9
10
K

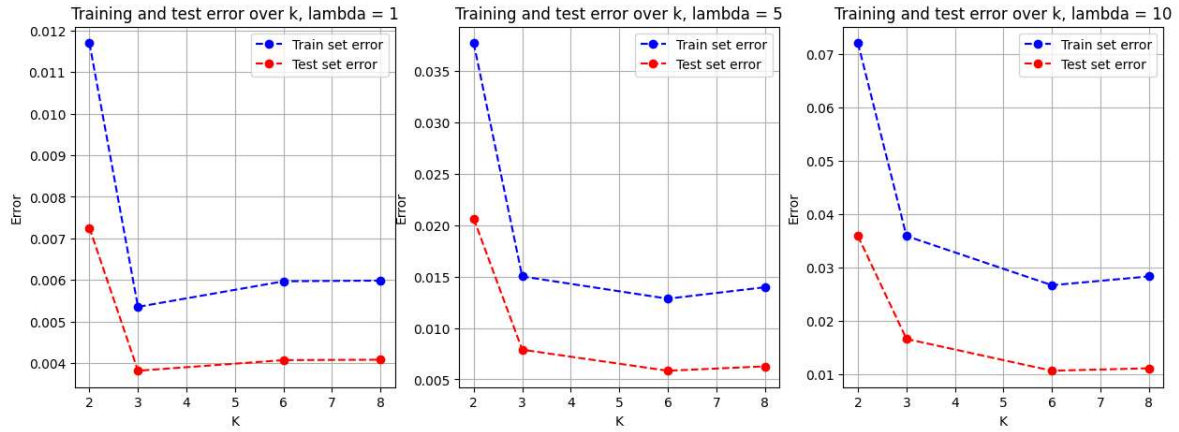


In [ ]:
```python
def grad_f_MAP(X, y, theta, lam):
    p = phi(X, len(theta))
    return p.T @ ((p @ theta) - y) + 2 * lam * theta

def theta_calc_eq_MAP(phi_X, Y, lam, K):
    first_fact = phi_X.T @ phi_X + (lam * np.eye(K))
    second_fact = phi_X.T @ Y
    try:
        L = scipy.linalg.cholesky(first_fact, lower = True)
        y = scipy.linalg.solve_triangular(L, b, lower = True)
        theta_found_normeq = scipy.linalg.solve_triangular(L.T, y)
    except:
        theta_found_normeq = np.linalg.solve(first_fact, second_fact)
    return theta_found_normeq

def MAP(D, K, lam, method):
    X, Y = D
    theta = None
    if method == "NE":
        theta = theta_calc_eq_MAP(phi(X, K), Y, lam, K)
    if method == "GD":
        w0 = np.random.normal(mean, var, K)
        theta = GD(w0, D, grad_f_MAP, lam=lam)
    if method == "SGD":
        w0 = np.random.normal(mean, var, K)
        theta = SGD(w0, D, grad_f_MAP, lam=lam)
    return theta
```

In [ ]:
```python
ks_MAP = [2, 3, 6, 8]
thetas_MAP = []
methods = ["NE", "NE", "NE", "NE"]
lams = [1, 5, 10]
```

```
errs_train_MAP = []
errs_test_MAP = []

for i, k in enumerate(ks_MAP):
    thetas_MAP_l = []
    errs_train_MAP_l = []
    errs_test_MAP_l = []
    for lam in lams:
        thetas_MAP_l.append(MAP((X_train, Y_train), k, lam, methods[i]))
        errs_train_MAP_l.append(avg_abs_err(thetas_MAP_l[-1], (X_train, Y_train)
        errs_test_MAP_l.append(avg_abs_err(thetas_MAP_l[-1], (X_test, Y_test)))
    thetas_MAP.append(thetas_MAP_l)
    errs_train_MAP.append(errs_train_MAP_l)
    errs_test_MAP.append(errs_test_MAP_l)

for i, k in enumerate(ks_MAP):
    plt.figure(figsize = (20, 5))
    plt.suptitle(f"K = {k}")
    for j, lam in enumerate(lams):
        theta_MAP = thetas_MAP[i][j]
        X_plot = np.linspace(a, b, 1000)
        Y_plot = phi(X_plot, k) @ theta_MAP
        plt.subplot(1, 4, j+1)
        plt.plot(X_plot, Y_plot, label=f"MAP with {methods[i]}", color="k")
        plt.scatter(X_train, Y_train, label= "Training Data", color='b')
        plt.scatter(X_test, Y_test, label="Test Data", color='r')
        plt.title(f"lambda = {lam}")
        plt.legend()
    plt.subplot(1, 4, 4)
    plt.plot(lams, errs_train_MAP[i], 'bo--', label="Train set error")
    plt.plot(lams, errs_test_MAP[i], 'ro--', label="Test set error")
    plt.legend()
    plt.title("Training and test error over lamda")
    plt.xlabel("lambda")
    plt.ylabel("Error")
    plt.show()

lam1errtrain = [l[0] for l in errs_train_MAP]
lam2errtrain = [l[1] for l in errs_train_MAP]
lam3errtrain = [l[2] for l in errs_train_MAP]
lam1errtest = [l[0] for l in errs_test_MAP]
lam2errtest = [l[1] for l in errs_test_MAP]
lam3errtest = [l[2] for l in errs_test_MAP]
lam_err_train_MAP = [lam1errtrain, lam2errtrain, lam3errtrain]
lam_err_test_MAP = [lam1errtest, lam2errtest, lam3errtest]
plt.figure(figsize = (15, 5))
for i, l in enumerate(lams):
    plt.subplot(1, 3, i+1)
    plt.plot(ks_MAP, lam_err_train_MAP[i], 'bo--', label="Train set error")
    plt.plot(ks_MAP, lam_err_test_MAP[i], 'ro--', label="Test set error")
    plt.grid()
    plt.legend()
    plt.title(f"Training and test error over k, lambda = {l}")
    plt.xlabel("K")
    plt.ylabel("Error")
```

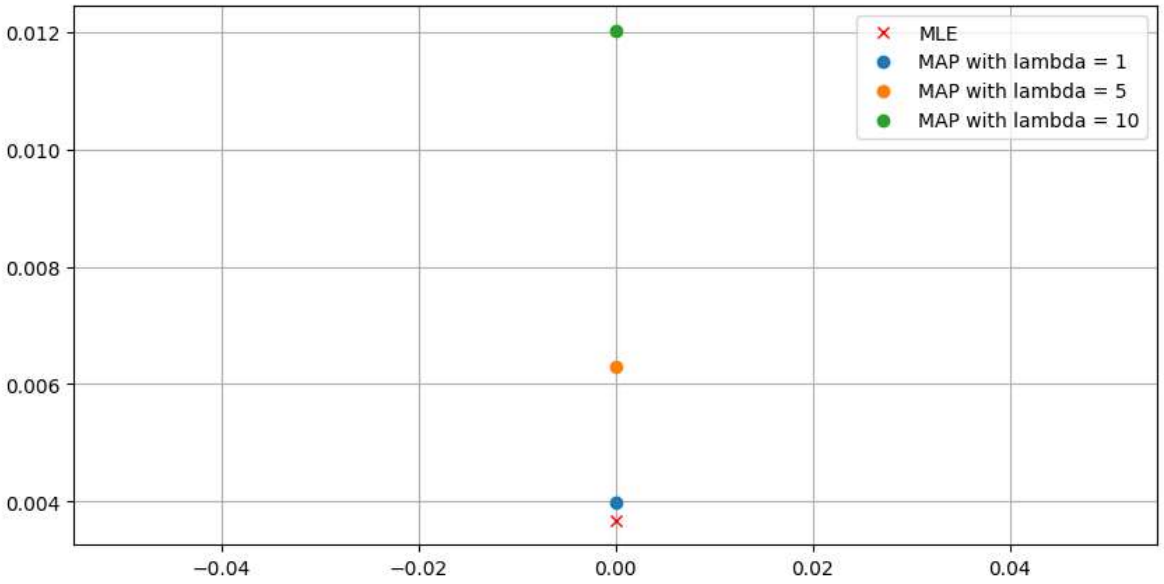

K = 2
lambda = 1
lambda = 5
lambda = 10
Training and test error over lamda
MAP with NE
Training Data
Test Data
Train set error
Test set error
Error
lambda
K = 3
K = 6
K = 8


Training and test error over k, lambda = 1
Training and test error over k, lambda = 5
Training and test error over k, lambda = 10
Train set error
Test set error
Error
K

In [ ]:
```
K_big = 26
```

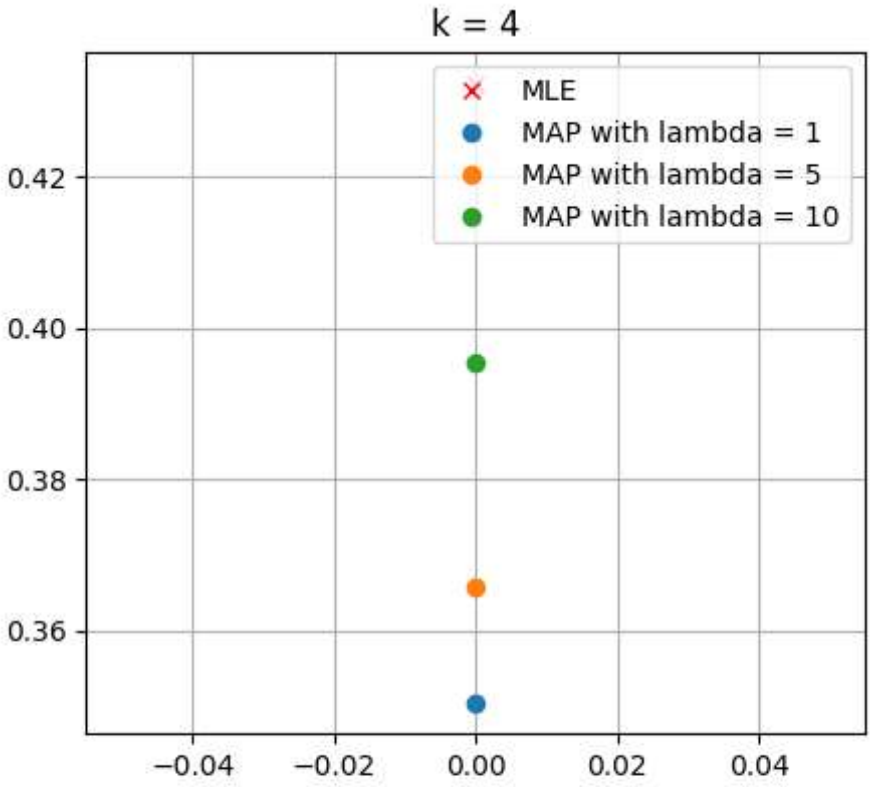
```
theta_MLE_big = MLE((X_train, Y_train), K_big, "NE")
thetas_MAP_big = [MAP((X_train, Y_train), K_big, lam, "NE") for lam in lams]

MLE_err = avg_abs_err(theta_MLE_big, (X_test, Y_test))
MAP_err = [avg_abs_err(theta, (X_test, Y_test)) for theta in thetas_MAP_big]

plt.figure(figsize = (10, 5))
plt.plot(MLE_err, 'xr', label="MLE")
for i, lam in enumerate(lams):
    plt.plot(MAP_err[i], 'o', label=f"MAP with lambda = {lam}")
plt.legend()
plt.grid()

print("MLE test error: ", MLE_err)
print("MAP test error: ", MAP_err)
```

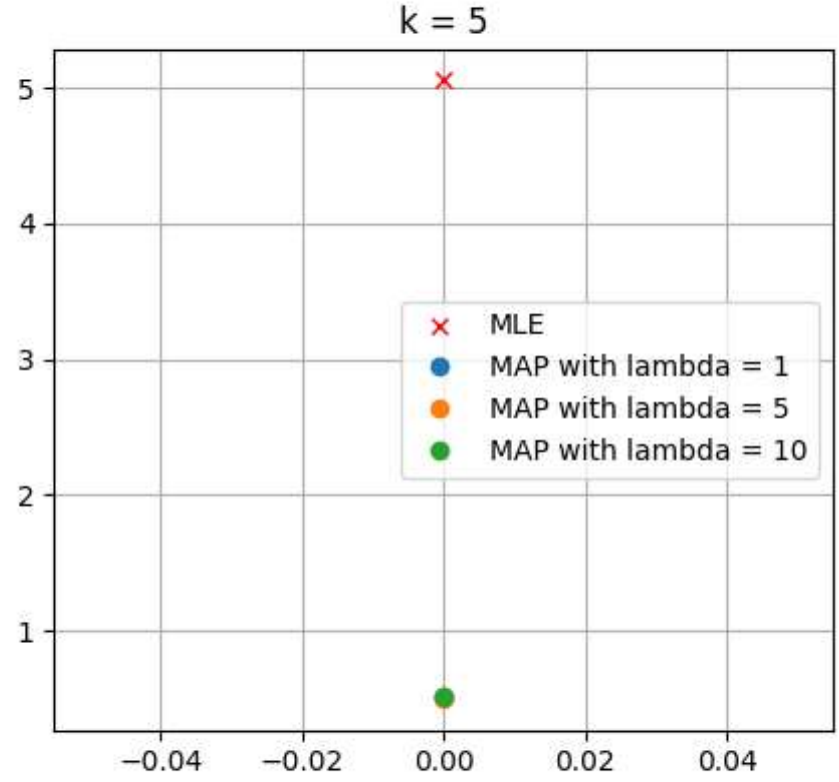
```
MLE test error:  0.003678542798268047
MAP test error:  [0.003977622155641768, 0.006305515191210238, 0.012028903745291558]
```



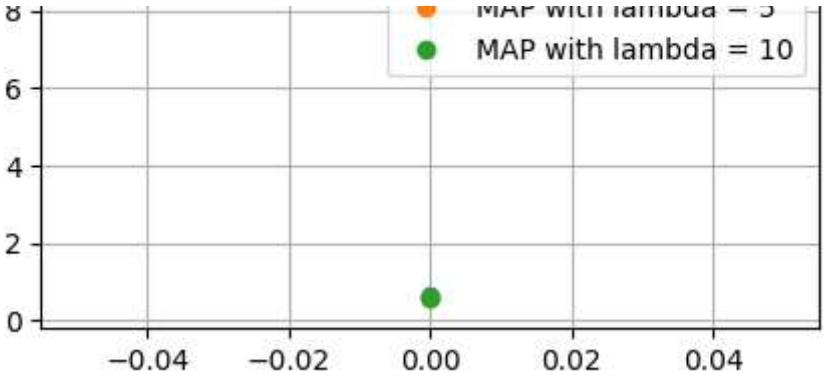
In [ ]:
```
def err(theta, k):
    theta_true = np.concatenate((np.ones((k, )), np.zeros((len(theta) - k,))))
    return ((np.linalg.norm(theta - theta_true)) / (np.linalg.norm(theta_true)))
```

In [ ]:
```
ks = [4, 5, 6]
lams = [1, 5, 10]

plt.figure(figsize=(5, 15))
for i, k in enumerate(ks):
    theta_MLE = MLE((X_train, Y_train), k, "NE")
    thetas_MAP = [MAP((X_train, Y_train), k, lam, "NE") for lam in lams]
    MLE_err = err(theta_MLE, k)
    MAP_err = [err(theta_MAP, k) for theta_MAP in thetas_MAP]

    plt.subplot(len(ks), 1, i+1)
    plt.plot(MLE_err, 'xr', label="MLE")

    for j, lam in enumerate(lams):
        plt.plot(MAP_err[j], 'o', label=f"MAP with lambda = {lam}")
    plt.legend()
    plt.grid()
    plt.title(f'k = {k}')
```


k = 4
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10
0.42
0.40
0.38
0.36
−0.04
−0.02
0.00
0.02
0.04


k = 5
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10
5
4
3
2
1
−0.04
−0.02
0.00
0.02
0.04


k = 6
MLE
MAP with lambda = 1
16
14
12
10



Varying train size

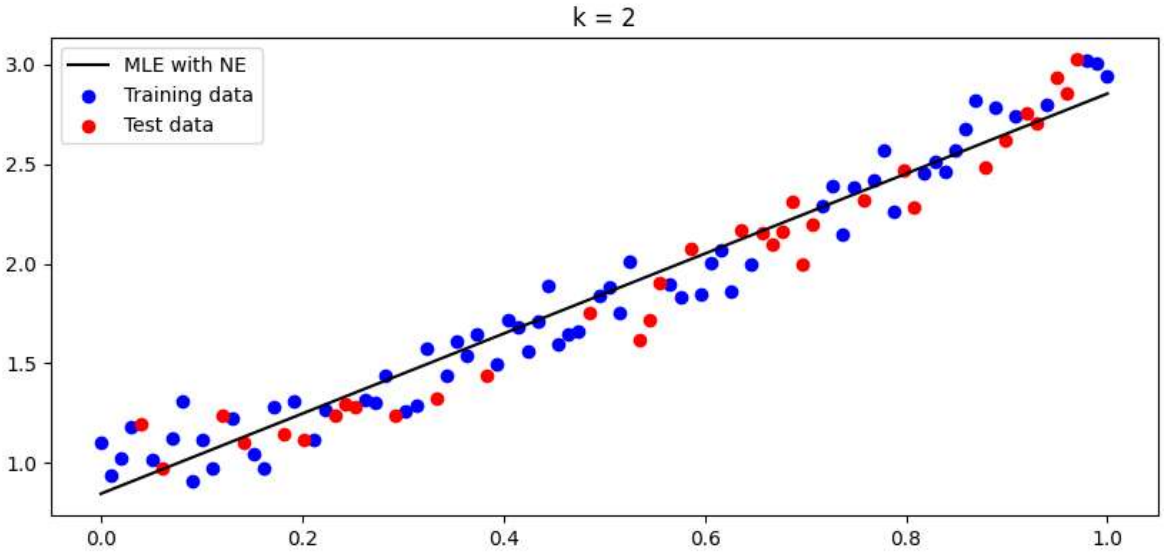
In [ ]:

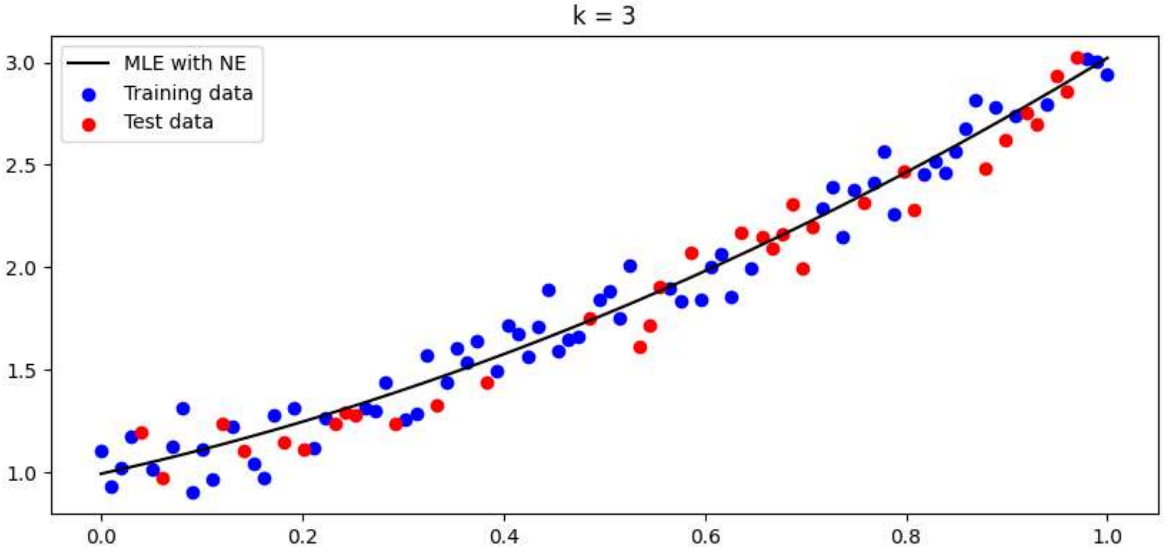
```
Ns = [100, 250, 500]
for N in Ns:
    print('\n')
    print('----------------------------------------')
    print('N =', N)
    print('\n')

    X = np.linspace(a, b, N)
    e = np.random.normal(0, var, N)
    Y = phi(X, K) @ theta_true + e
    D = (X, Y)
    N_train = int(N / 3 * 2)
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, N_train)

    ks_MLE = [2, 3, 6, 10]
    methods = [['NE', 'NE', 'NE', 'NE'],
               ['GD', 'GD', 'GD', 'GD'],
               ['SGD', 'SGD', 'SGD', 'SGD']]

    for j in range(len(methods)):
        print('#### Method:', methods[j][0], '####')
        thetas_MLE = []
        errs_train_MLE = []
        errs_test_MLE = []
        for i, k in enumerate(ks_MLE):
            thetas_MLE.append(MLE((X_train, Y_train), k, methods[j][i]))
            errs_train_MLE.append(avg_abs_err(thetas_MLE[-1], (X_train, Y_train)
            errs_test_MLE.append(avg_abs_err(thetas_MLE[-1], (X_test, Y_test)))

        fig, ax = plt.subplots(len(ks_MLE), figsize=(10, 20))

        for i in range(len(ks_MLE)):
            X_plot = np.linspace(a, b, 1000)
            Y_plot = phi(X_plot, ks_MLE[i]) @ thetas_MLE[i]
            ax[i].plot(X_plot, Y_plot, label=f'MLE with {methods[j][i]}', color=
            ax[i].scatter(X_train, Y_train, label='Training data', color='b')
            ax[i].scatter(X_test, Y_test, label='Test data', color='r')
            ax[i].set_title('k = ' + str(ks_MLE[i]))
            ax[i].legend()
        plt.show()

        plt.plot(ks_MLE, errs_train_MLE, label="Train set error", color='b')
        plt.plot(ks_MLE, errs_test_MLE, label="Test set error", color='r')
```

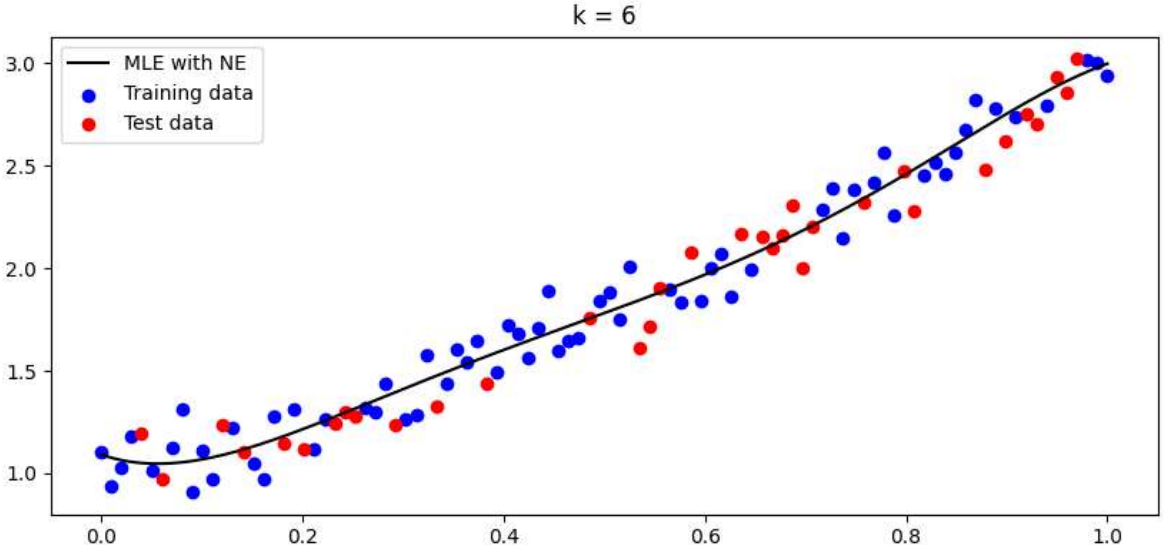
```
        plt.legend()
        plt.title("Training and test error over k")
        plt.xlabel("K")
        plt.ylabel("Error")
        plt.show()

    ks_MAP = [2, 3, 6, 8]
    thetas_MAP = []
    lams = [1, 5, 10]

    for j in range(len(methods)):
        print('#### Method:', methods[j][0], '####')
        thetas_MAP = []
        for i, k in enumerate(ks_MAP):
            thetas_MAP_l = []
            for lam in lams:
                thetas_MAP_l.append(MAP((X_train, Y_train), k, lam, methods[j][i
            thetas_MAP.append(thetas_MAP_l)

        for i, k in enumerate(ks_MAP):
            plt.figure(figsize = (20, 5))
            plt.suptitle(f"K = {k}")
            for h, lam in enumerate(lams):
                theta_MAP = thetas_MAP[i][h]
                X_plot = np.linspace(a, b, 1000)
                Y_plot = phi(X_plot, k) @ theta_MAP
                plt.subplot(1, 3, h+1)
                plt.plot(X_plot, Y_plot, label=f"MAP with {methods[j][i]}", colo
                plt.scatter(X_train, Y_train, label= "Training Data", color='b')
                plt.scatter(X_test, Y_test, label="Test Data", color='r')
                plt.title(f"lambda = {lam}")
                plt.legend()
        plt.show()

    testMLEGD = MLE((X_train, Y_train), K, "GD")
    errMLEGD = err(testMLEGD, K)
    print(f"Theta MLE GD: {testMLEGD}")
    print(f"Error MLE GD: {errMLEGD}")

    testMLESGD = MLE((X_train, Y_train), K, "SGD")
    errMLESGD = err(testMLESGD, K)
    print(f"Theta MLE SGD: {testMLESGD}")
    print(f"Error MLE SGD: {errMLEGD}")

    testMLENE = MLE((X_train, Y_train), K, "NE")
    errMLENE = err(testMLENE, K)
    print(f"Theta MLE NE: {testMLENE}")
    print(f"Error MLE NE: {errMLENE}")

    print()

    testMAPGD = MAP((X_train, Y_train), K, 1, "GD")
    errMAPGD = err(testMAPGD, K)
    print(f"Theta MAP GD: {testMAPGD}")
    print(f"Error MAP GD: {errMAPGD}")

    testMAPSGD = MAP((X_train, Y_train), K, 1, "SGD")
    errMAPSGD = err(testMAPSGD, K)
    print(f"Theta MAP SGD: {testMAPSGD}")
    print(f"Error MAP SGD: {errMAPGD}")
```

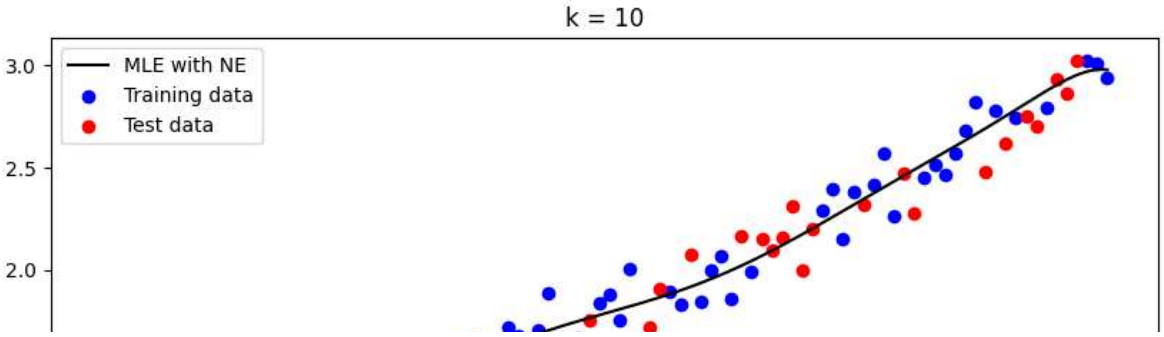
```
    testMAPNE = MAP((X_train, Y_train), K, 1, "NE")
    errMAPNE = err(testMAPNE, K)
    print(f"Theta MAP NE: {testMAPNE}")
    print(f"Error MAP NE: {errMAPNE}")
    print('\n')

    K_big = 26
    meths = ['NE', 'GD', 'SGD']
    for m in meths:
        print('Method:', m)
        print('\n')
        theta_MLE_big = MLE((X_train, Y_train), K_big, m)
        thetas_MAP_big = [MAP((X_train, Y_train), K_big, lam, m) for lam in lams

        MLE_err = avg_abs_err(theta_MLE_big, (X_test, Y_test))
        MAP_err = [avg_abs_err(theta, (X_test, Y_test)) for theta in thetas_MAP_
        print("MLE test error: ", MLE_err)
        print("MAP test error: ", MAP_err)

        plt.figure(figsize = (10, 5))
        plt.plot(MLE_err, 'xr', label="MLE")
        for i, lam in enumerate(lams):
            plt.plot(MAP_err[i], 'o', label=f"MAP with lambda = {lam}")
        plt.legend()
        plt.grid()
        plt.show()

        ks = [4, 5, 6]
        lams = [1, 5, 10]

        plt.figure(figsize=(5, 15))
        for i, k in enumerate(ks):
            theta_MLE = MLE((X_train, Y_train), k, m)
            thetas_MAP = [MAP((X_train, Y_train), k, lam, m) for lam in lams]
            MLE_err = err(theta_MLE, k)
            MAP_err = [err(theta_MAP, k) for theta_MAP in thetas_MAP]

            plt.subplot(len(ks), 1, i+1)
            plt.plot(MLE_err, 'xr', label="MLE")

            for j, lam in enumerate(lams):
                plt.plot(MAP_err[j], 'o', label=f"MAP with lambda = {lam}")
            plt.legend()
            plt.grid()
            plt.title(f'k = {k}')
        plt.show()
```

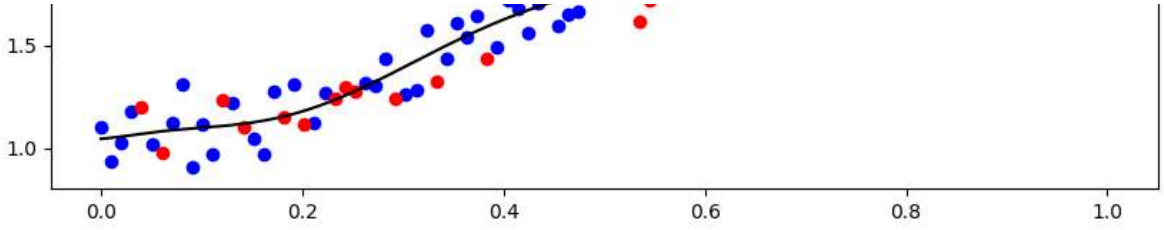
```
----------------------------------------
N = 100


#### Method: NE ####
```


k = 2
MLE with NE
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0


k = 3
MLE with NE
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0

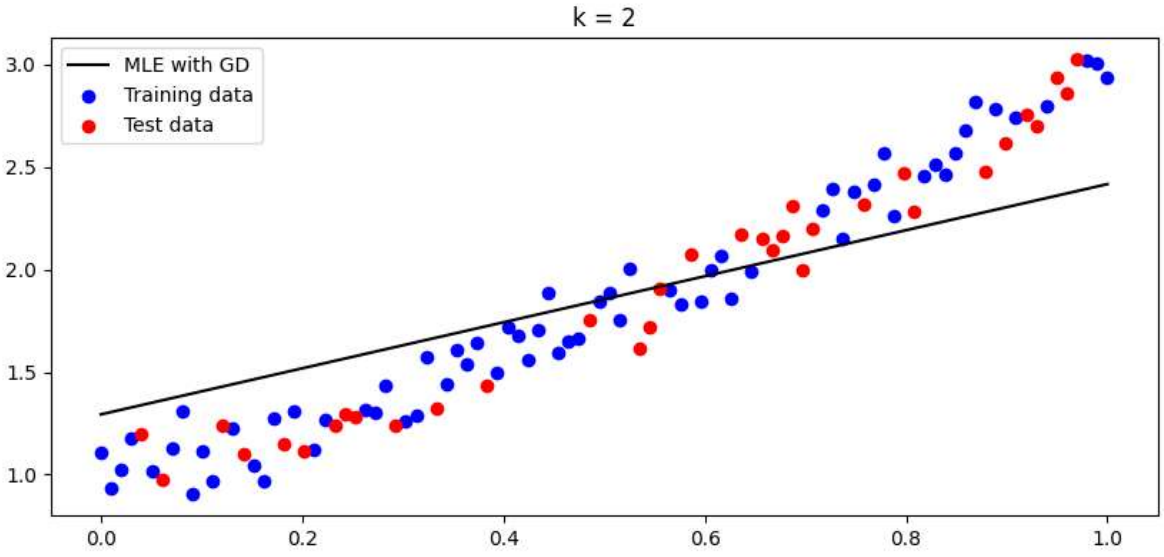

k = 6
MLE with NE
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0

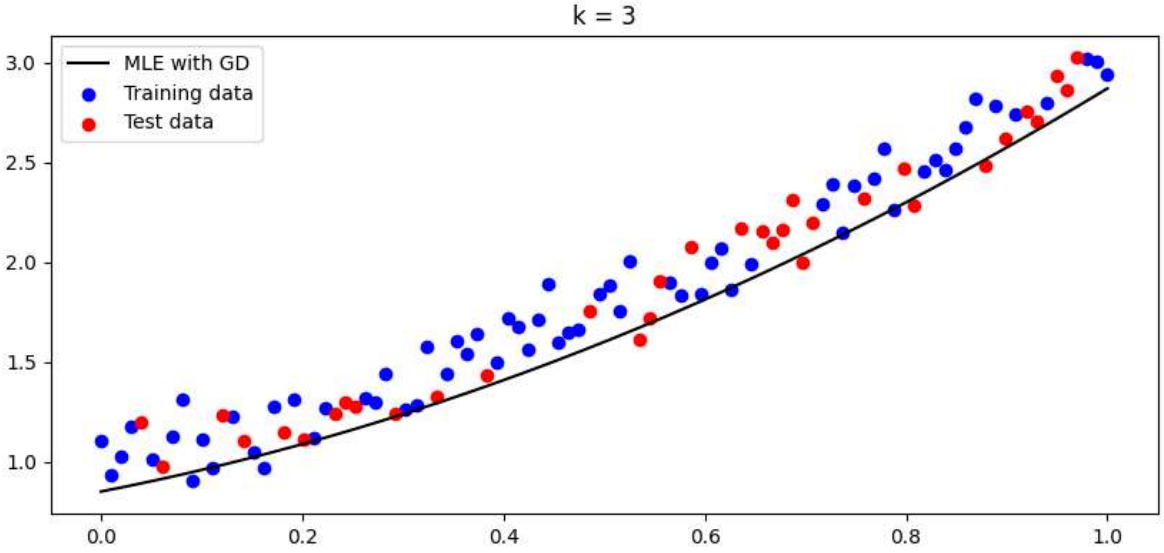

k = 10
MLE with NE
Training data
Test data
3.0
2.5
2.0

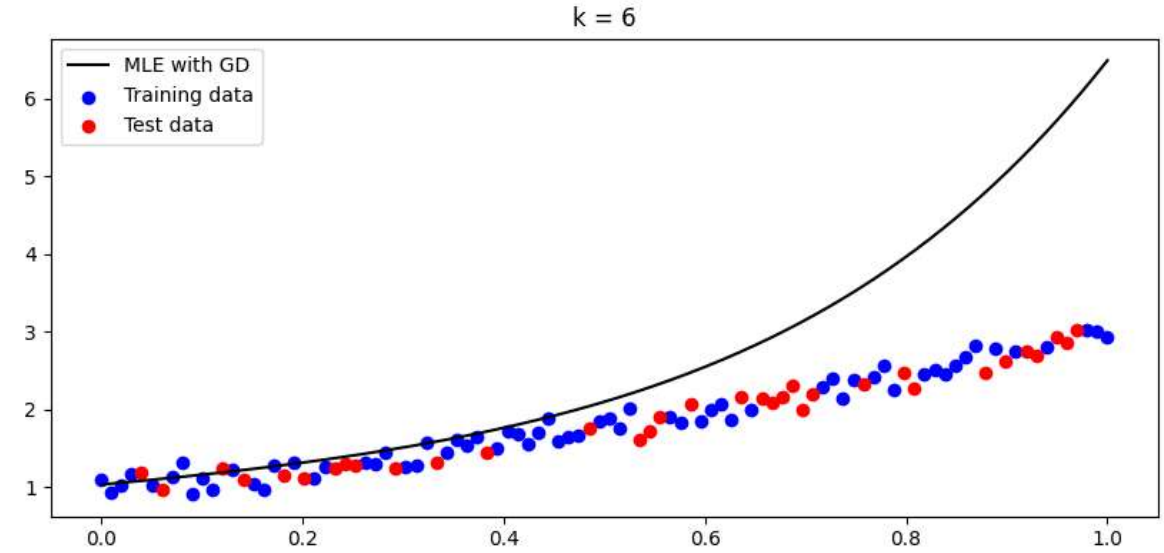

1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0

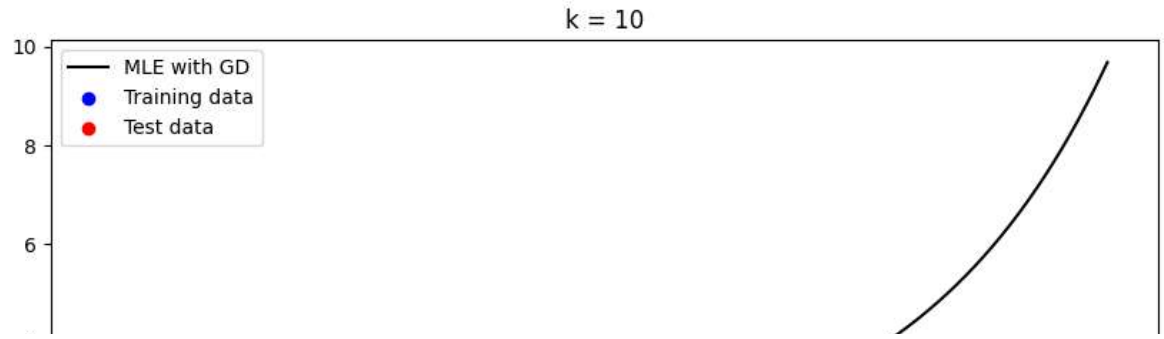

Training and test error over k
Train set error
Test set error
Error
0.011
0.010
0.009
0.008
0.007
0.006
0.005
0.004
2
3
4
5
6
7
8
9
10
K

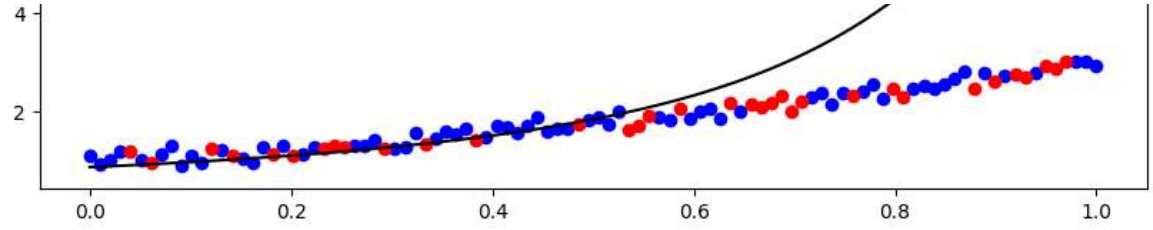
```
#### Method: GD ####
```


k = 2
MLE with GD
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0


k = 3
MLE with GD
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0

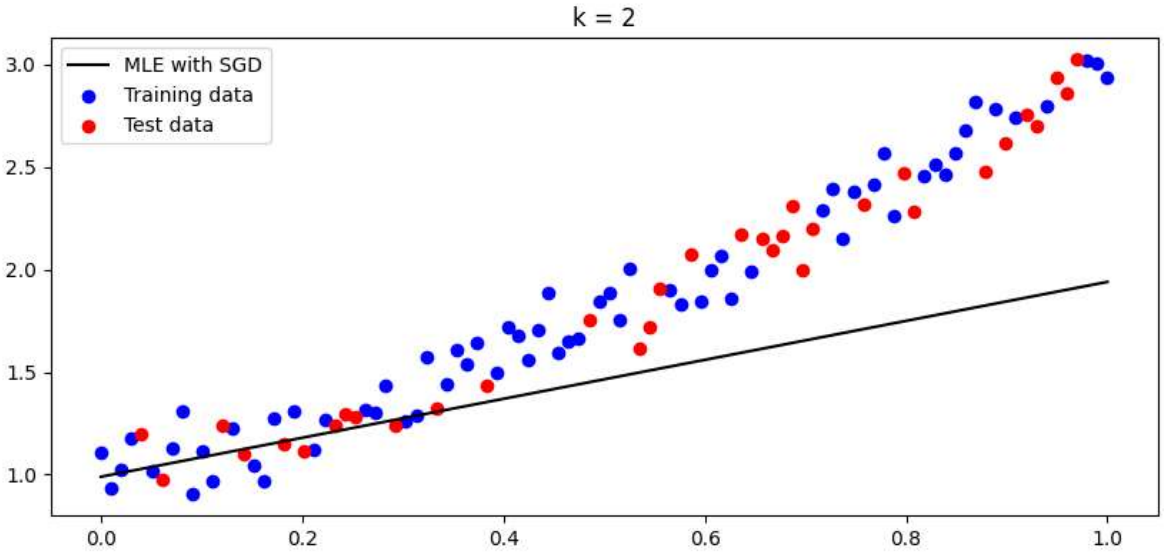

k = 6
MLE with GD
Training data
Test data
6
5
4
3
2
1
0.0
0.2
0.4
0.6
0.8
1.0

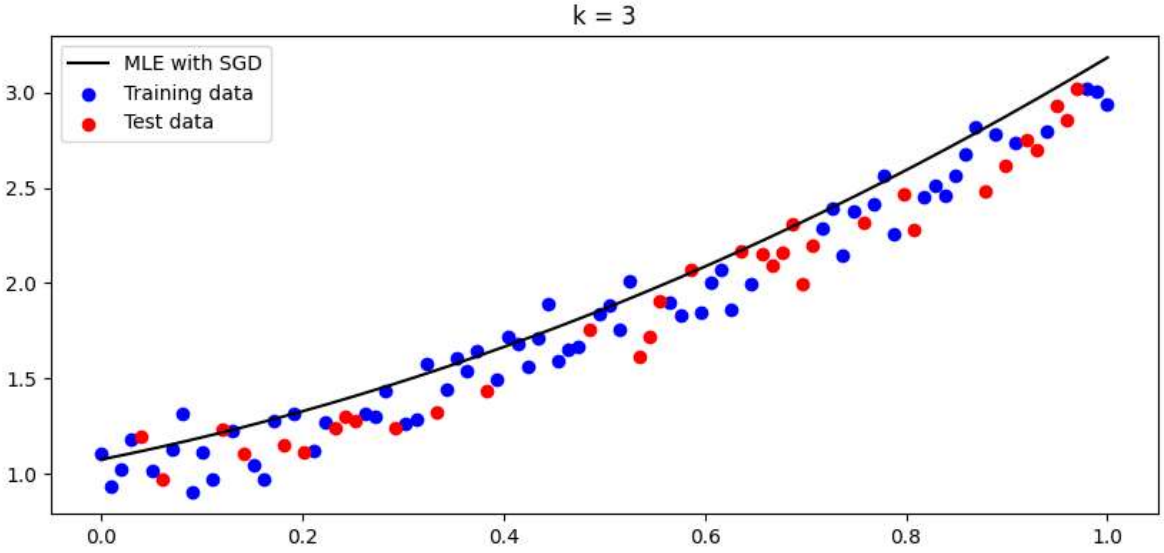

k = 10
MLE with GD
Training data
Test data
10
8
6

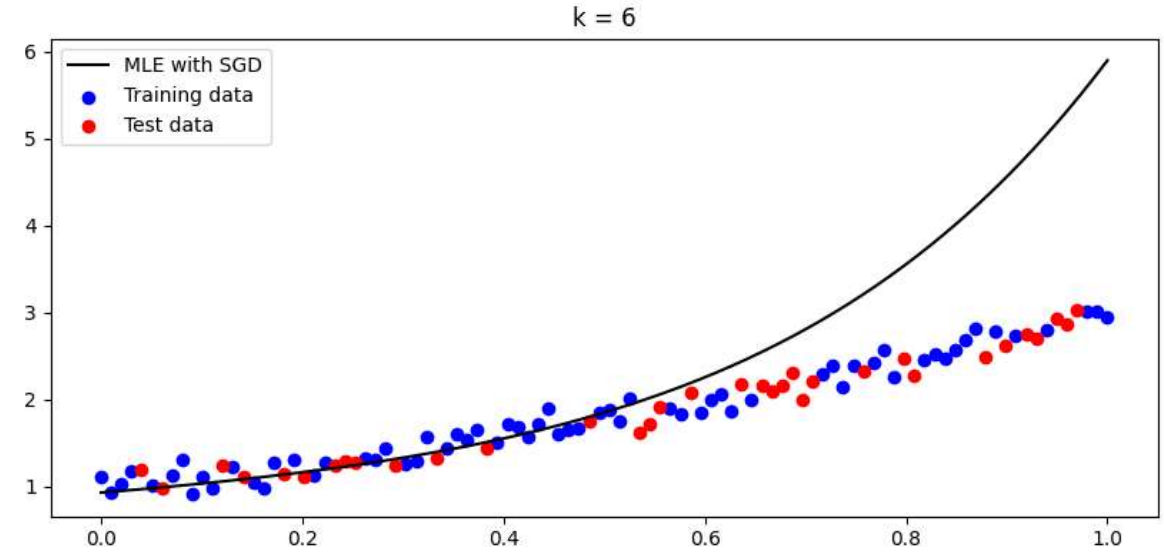

4
2
0.0
0.2
0.4
0.6
0.8
1.0

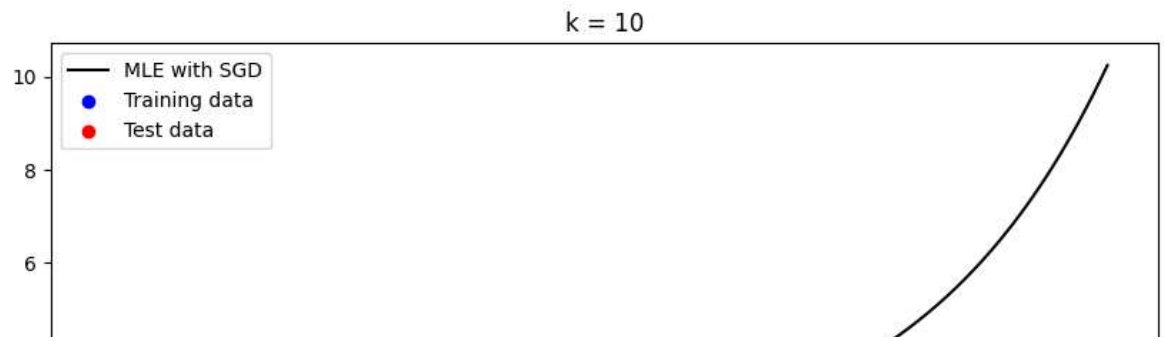

Training and test error over k
Train set error
Test set error
Error
K
2.0
1.5
1.0
0.5
0.0
2
3
4
5
6
7
8
9
10

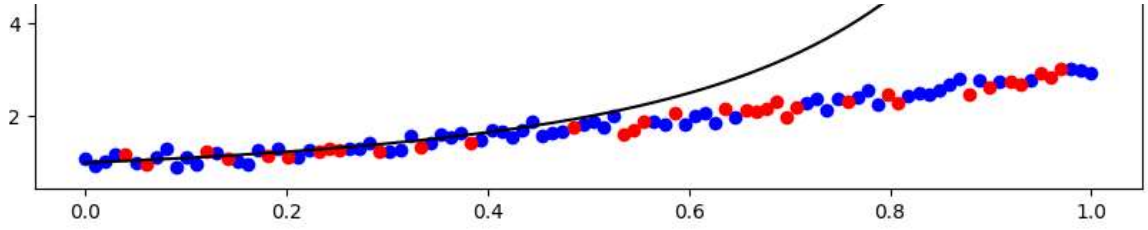
Method: SGD


k = 2
MLE with SGD
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0


k = 3
MLE with SGD
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0

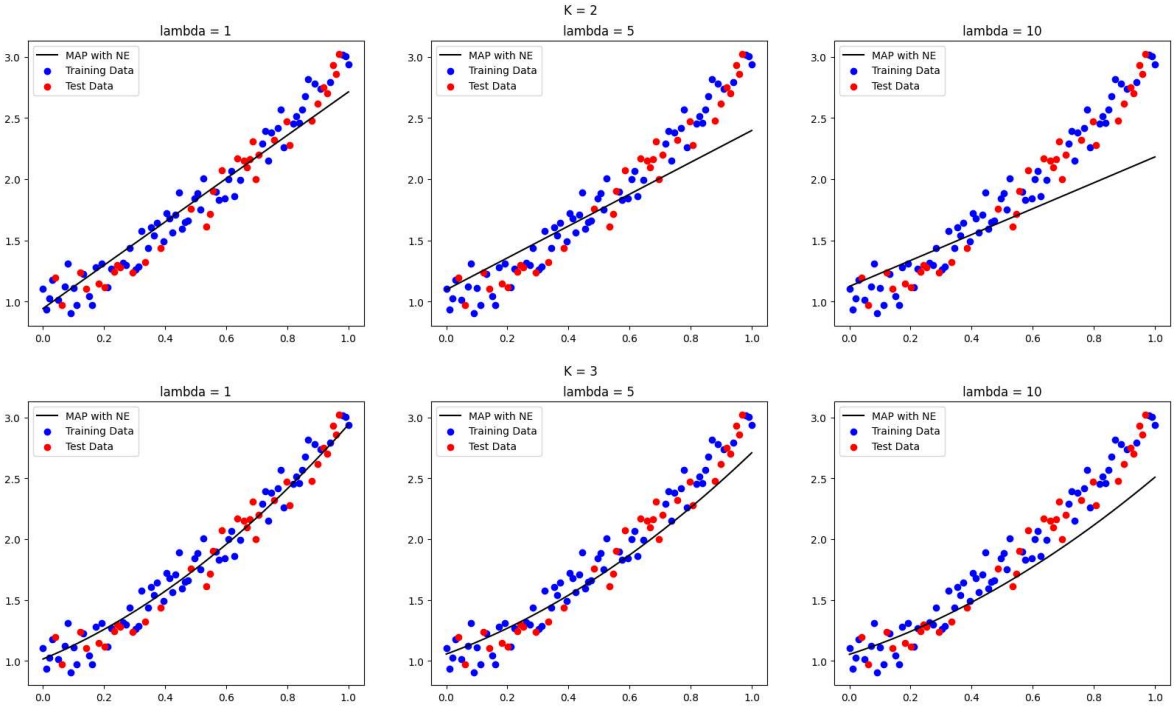

k = 6
MLE with SGD
Training data
Test data
6
5
4
3
2
1
0.0
0.2
0.4
0.6
0.8
1.0


k = 10
MLE with SGD
Training data
Test data
10
8
6

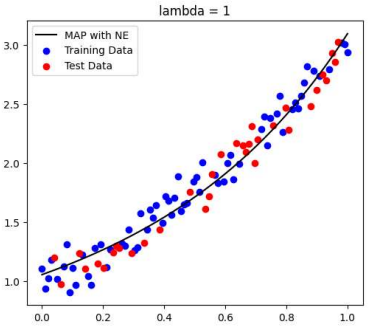

4
2
0.0
0.2
0.4
0.6
0.8
1.0

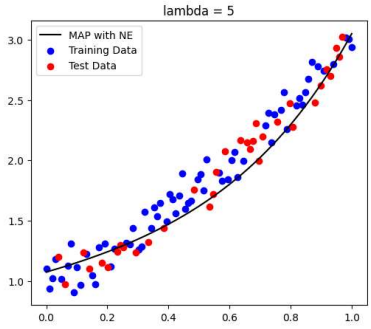

Training and test error over k
Train set error
Test set error
Error
K
2.5
2.0
1.5
1.0
0.5
0.0
2
3
4
5
6
7
8
9
10

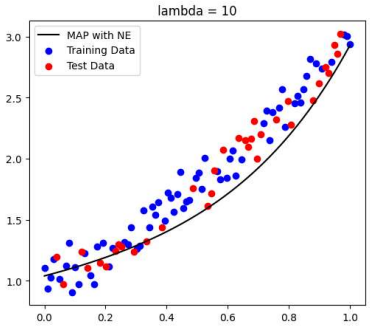
Method: NE


K = 2
lambda = 1
lambda = 5
lambda = 10
K = 3
lambda = 1
lambda = 5
lambda = 10
MAP with NE
Training Data
Test Data

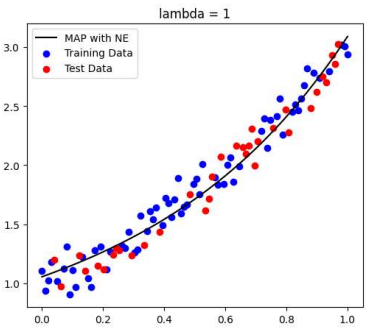
K = 6

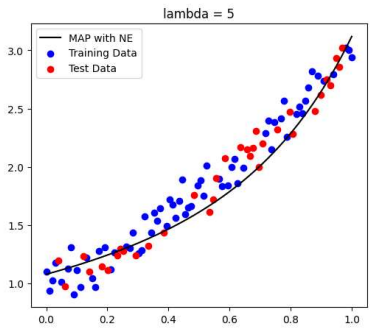


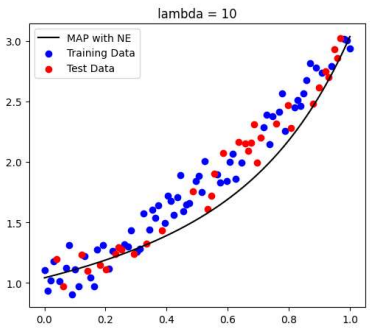




K = 8

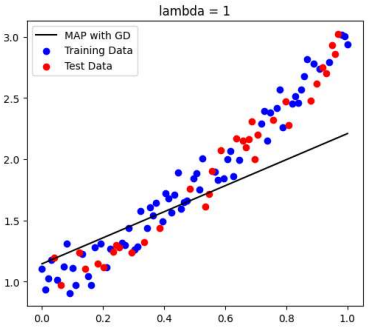


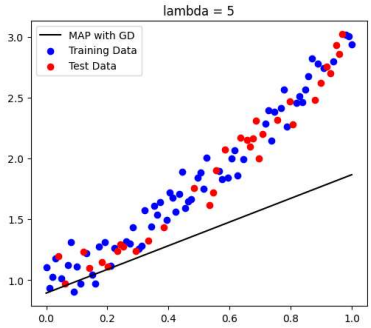


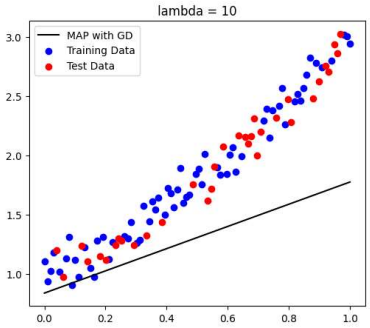


Method: GD

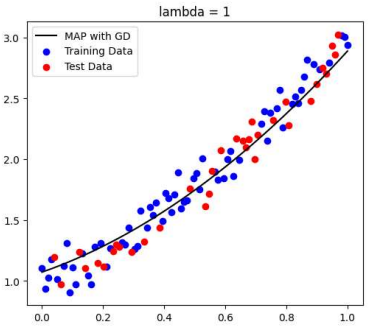
K = 2

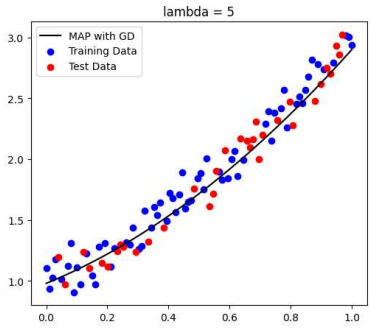


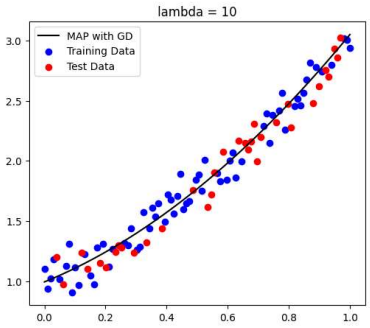




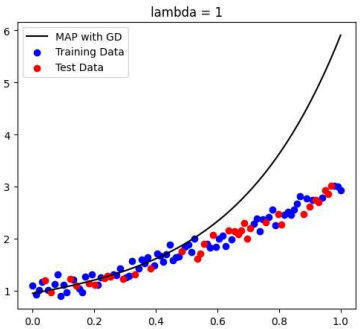
K = 3

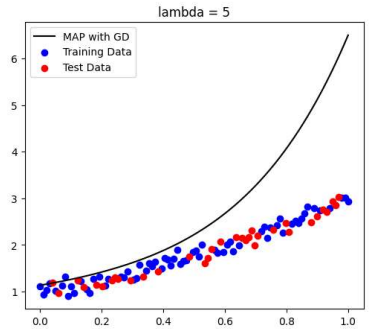


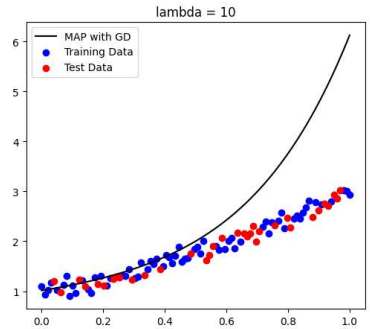




K = 6

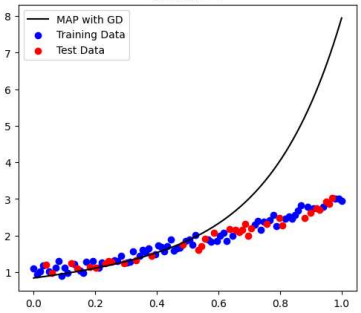




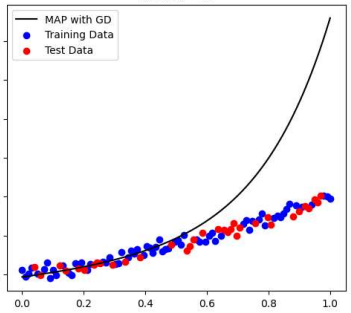


K = 8

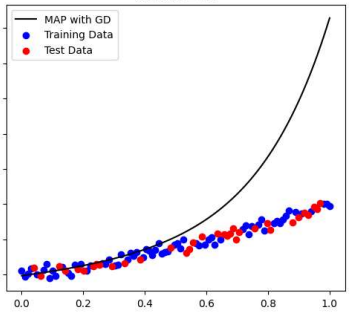
lambda = 1



lambda = 5



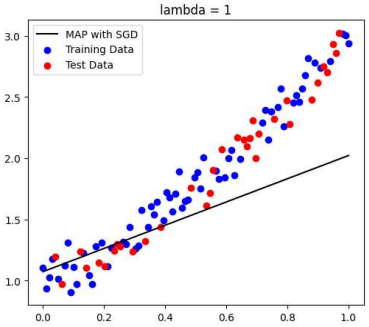
lambda = 10



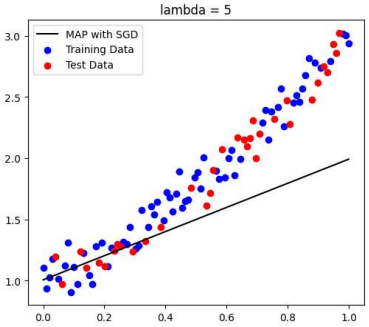
Method: SGD

K = 2

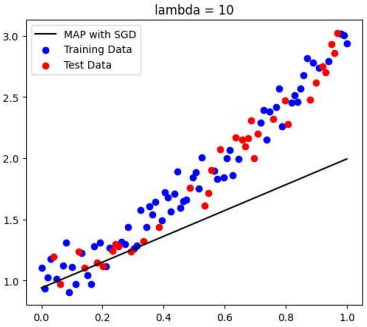
lambda = 1



lambda = 5

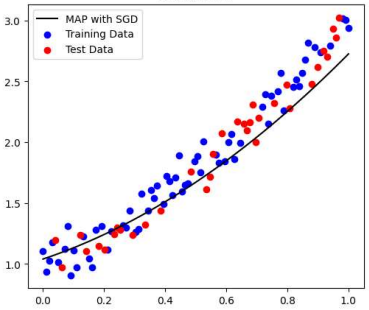


lambda = 10

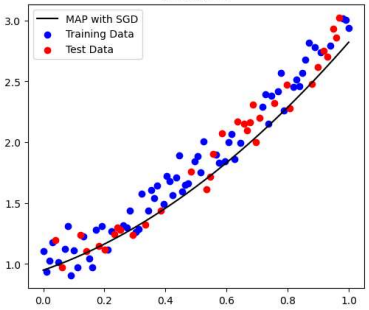


K = 3

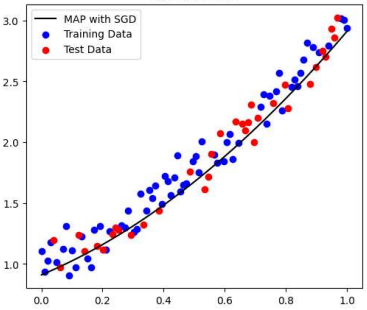
lambda = 1



lambda = 5

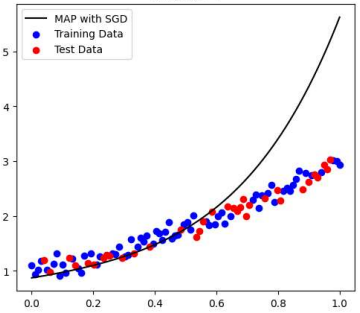


lambda = 10

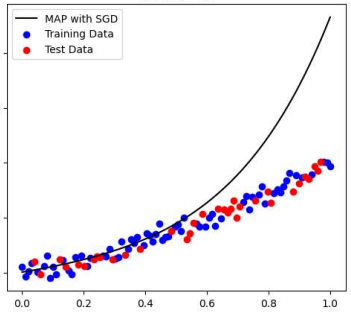


K = 6

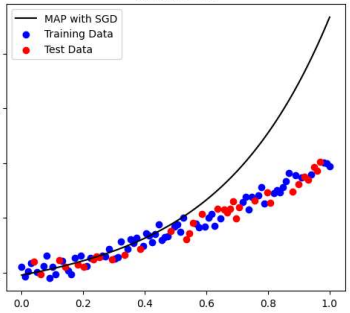
lambda = 1



lambda = 5

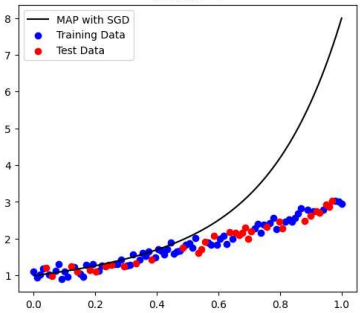


lambda = 10

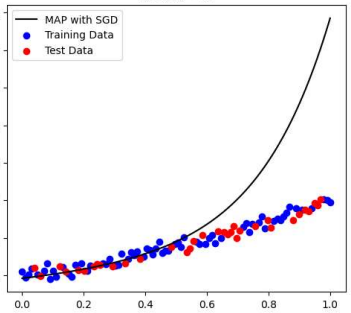


K = 8

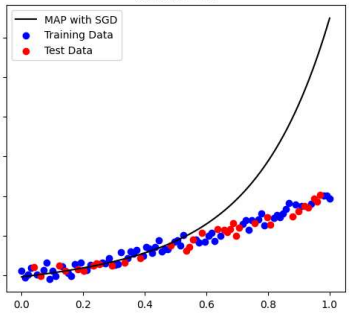
lambda = 1



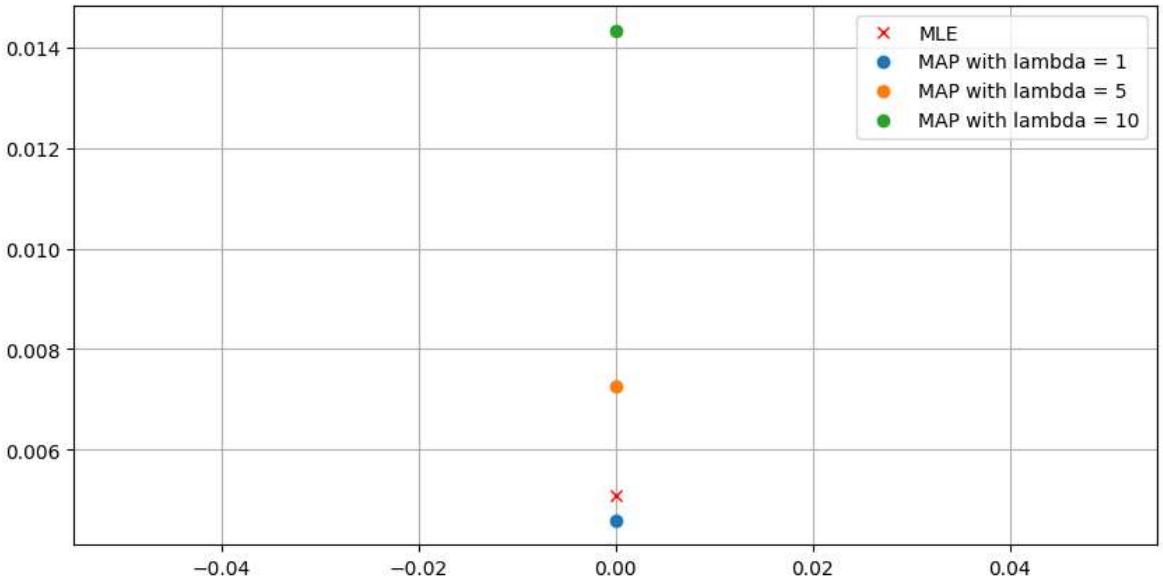
lambda = 5

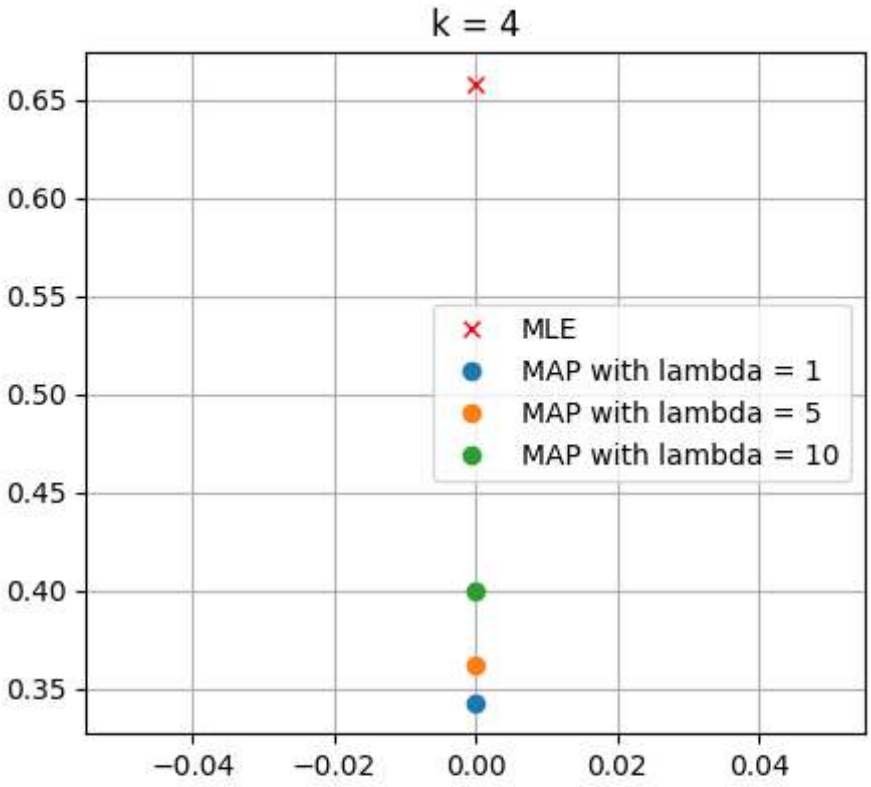


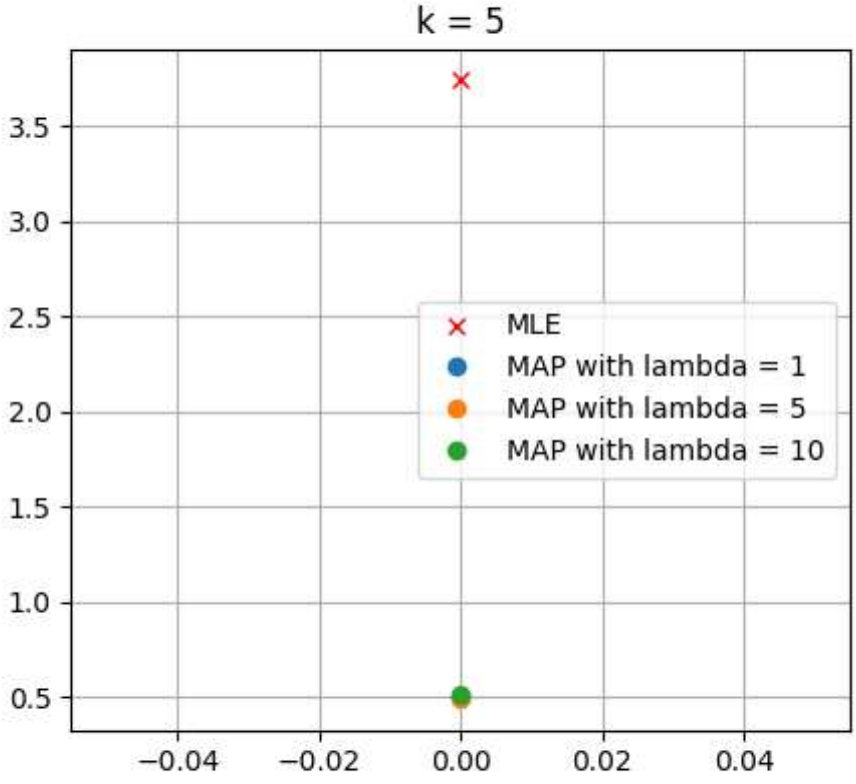
lambda = 10



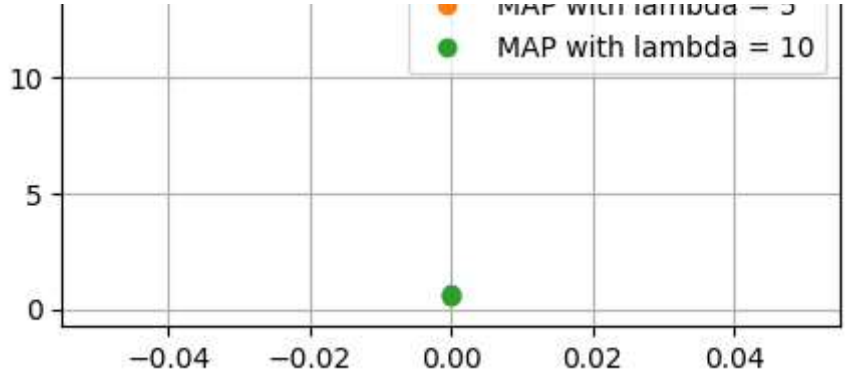
```
Theta MLE GD: [0.94543979 0.91695005 0.98314342]
Error MLE GD: 0.05819007471869636
Theta MLE SGD: [1.05338214 0.95425737 1.0097587 ]
Error MLE SGD: 0.05819007471869636
Theta MLE NE: [0.99337714 1.07676201 0.95131568]
Error MLE NE: 0.0526194881269128

Theta MAP GD: [0.96114916 1.00280544 1.01826159]
Error MAP GD: 0.024837768137234748
Theta MAP SGD: [1.10507766 0.94117245 0.91788544]
Error MAP SGD: 0.024837768137234748
Theta MAP NE: [1.01568313 1.0395981  0.88585177]
Error MAP NE: 0.07034151702760777


Method: NE


MLE test error:  0.005094156305275127
MAP test error:  [0.004595553746760605, 0.007250738160413078, 0.014338627080455375]
```




k = 4
0.65
0.60
0.55
0.50
0.45
0.40
0.35
−0.04
−0.02
0.00
0.02
0.04
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10

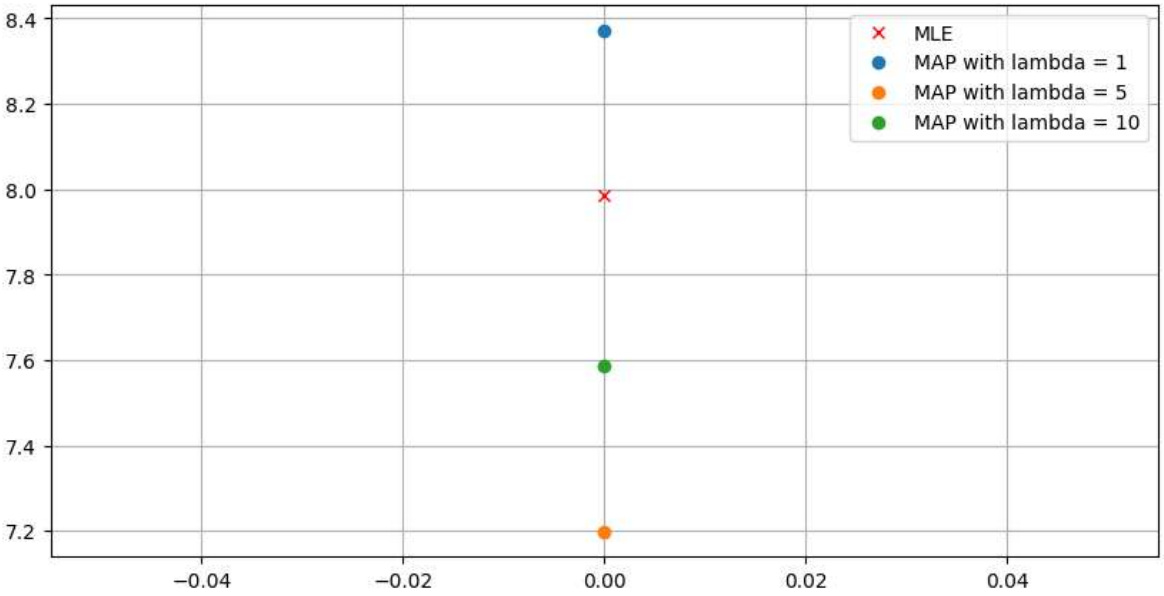

k = 5
3.5
3.0
2.5
2.0
1.5
1.0
0.5
−0.04
−0.02
0.00
0.02
0.04
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10

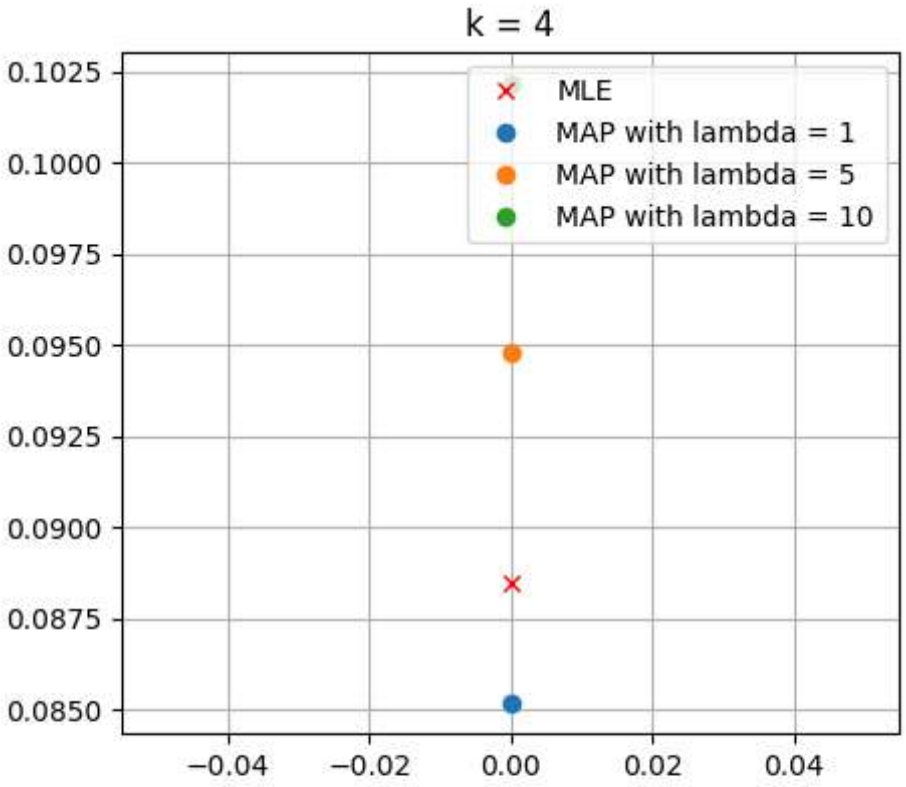

k = 6
25
20
15
MLE
MAP with lambda = 1

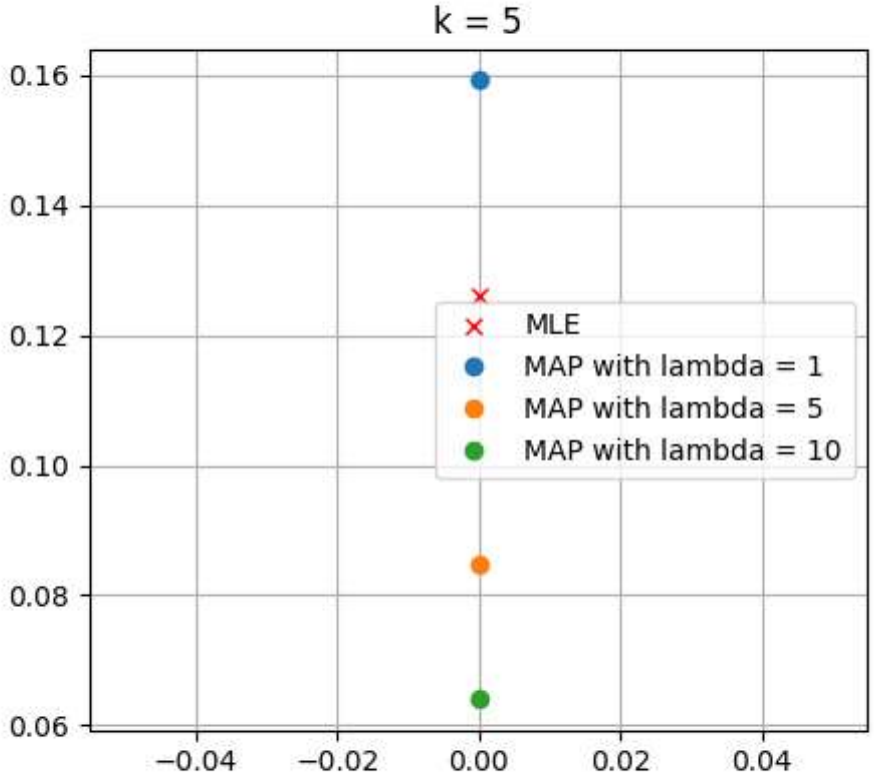


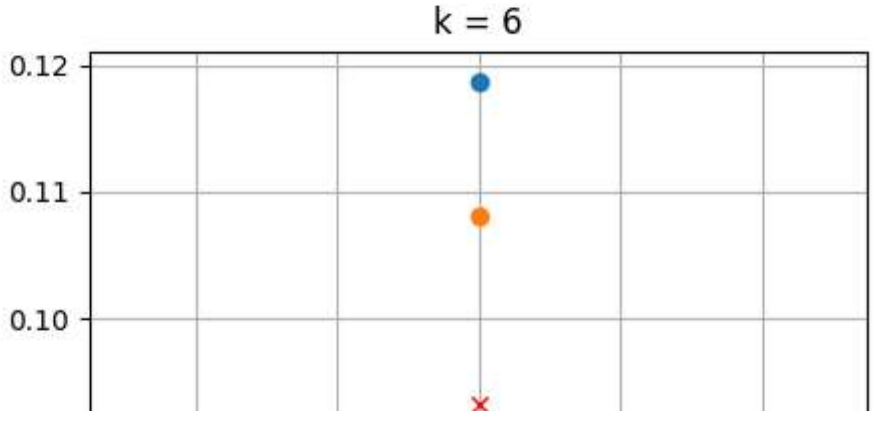
Method: GD

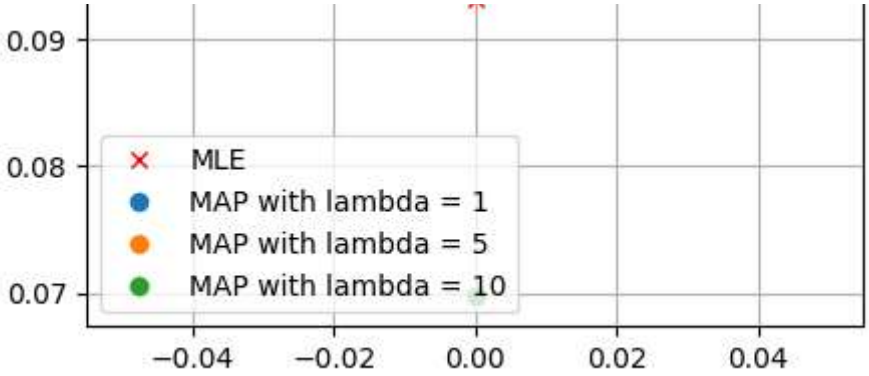
MLE test error: 7.986681084056094
MAP test error: [8.371862848539124, 7.198268130933647, 7.586937713064495]




k = 4
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10
0.1025
0.1000
0.0975
0.0950
0.0925
0.0900
0.0875
0.0850
−0.04
−0.02
0.00
0.02
0.04

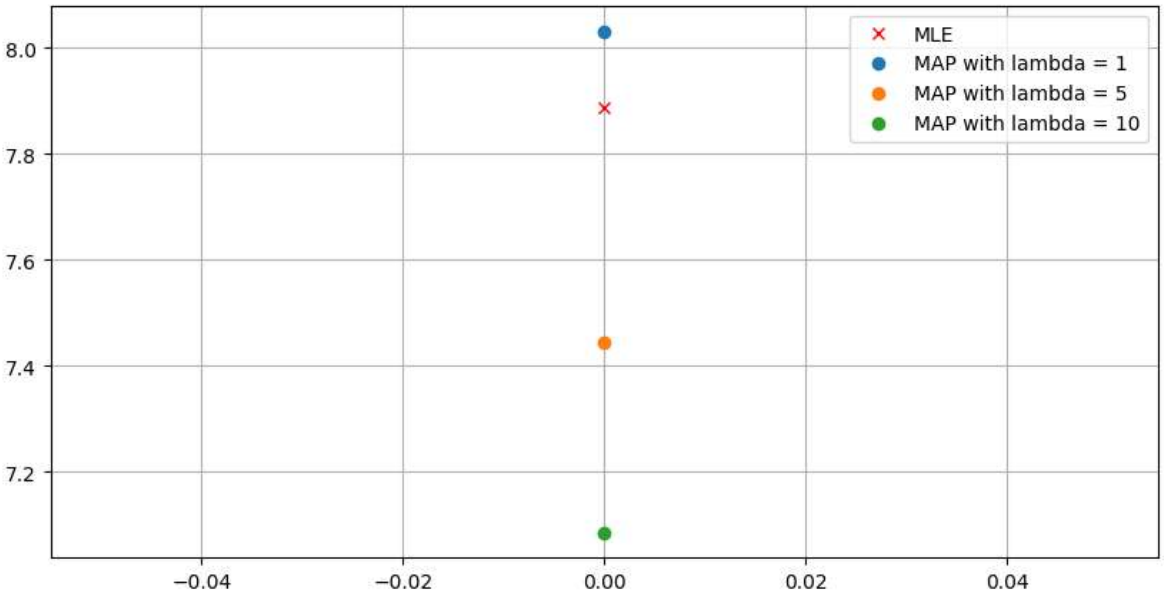

k = 5
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10
0.16
0.14
0.12
0.10
0.08
0.06
−0.04
−0.02
0.00
0.02
0.04

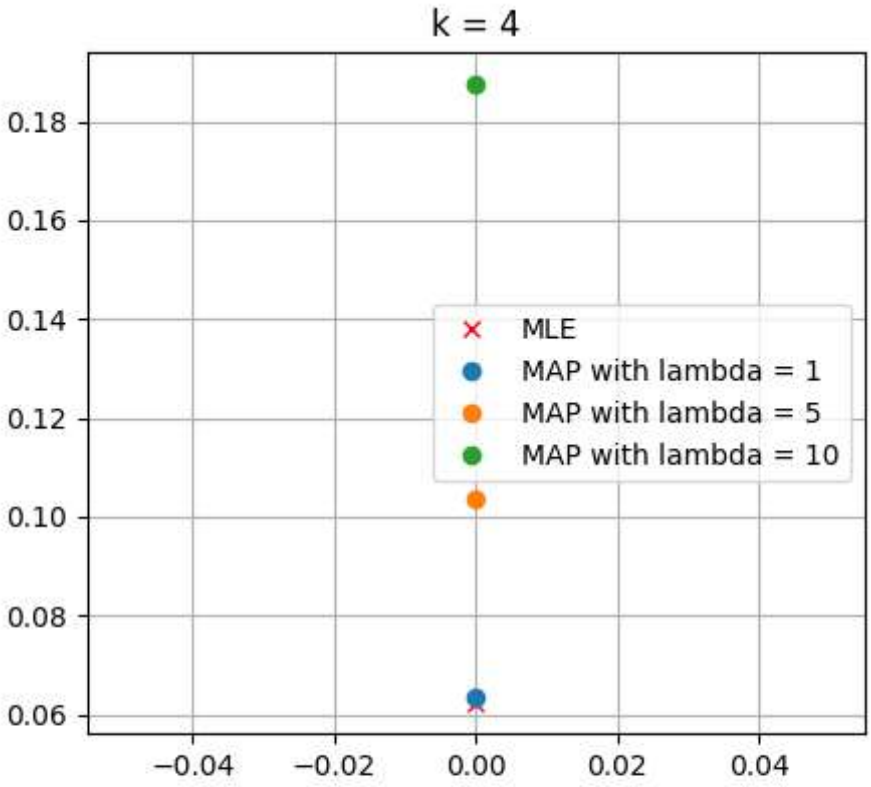

k = 6
0.12
0.11
0.10

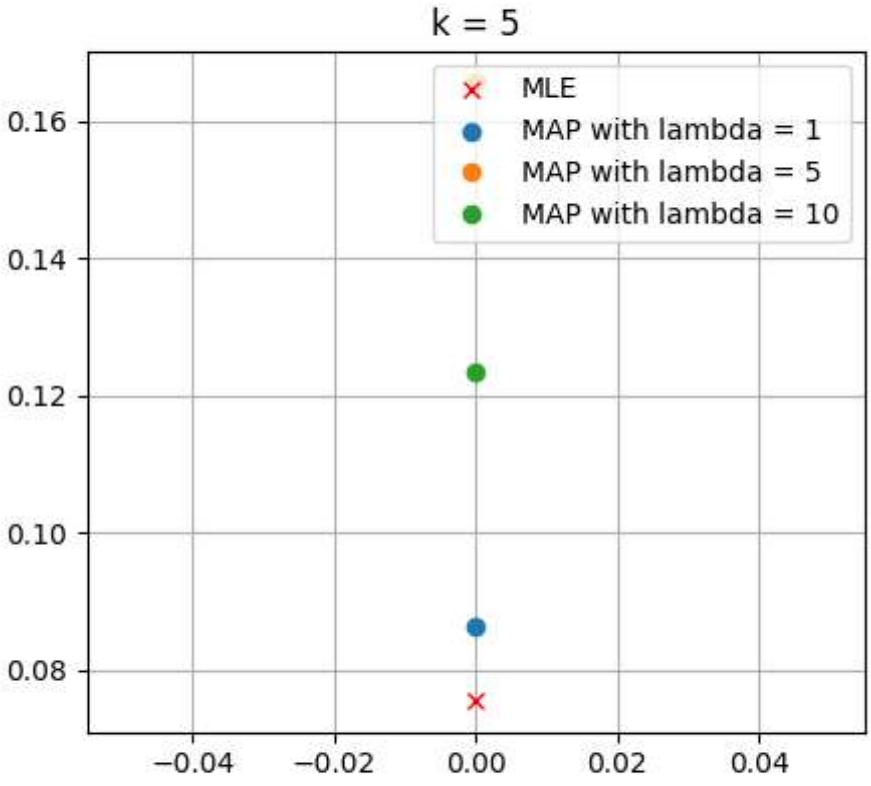


Method: SGD

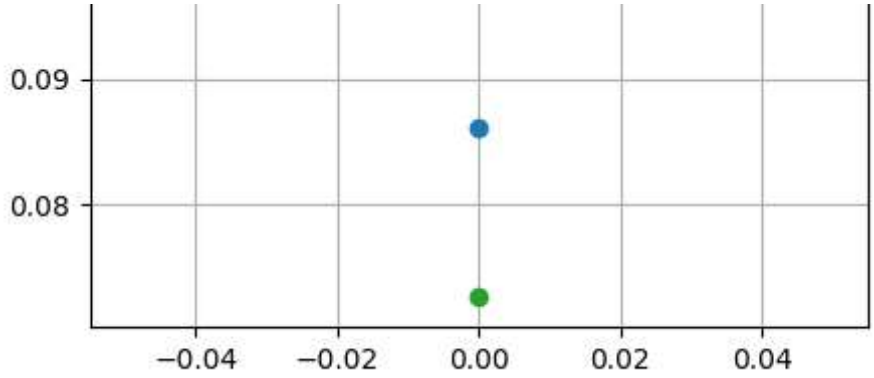
MLE test error: 7.887815851811448
MAP test error: [8.03111078284531, 7.444200950842138, 7.085722128666161]




k = 4
0.18
0.16
0.14
0.12
0.10
0.08
0.06
−0.04
−0.02
0.00
0.02
0.04
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10

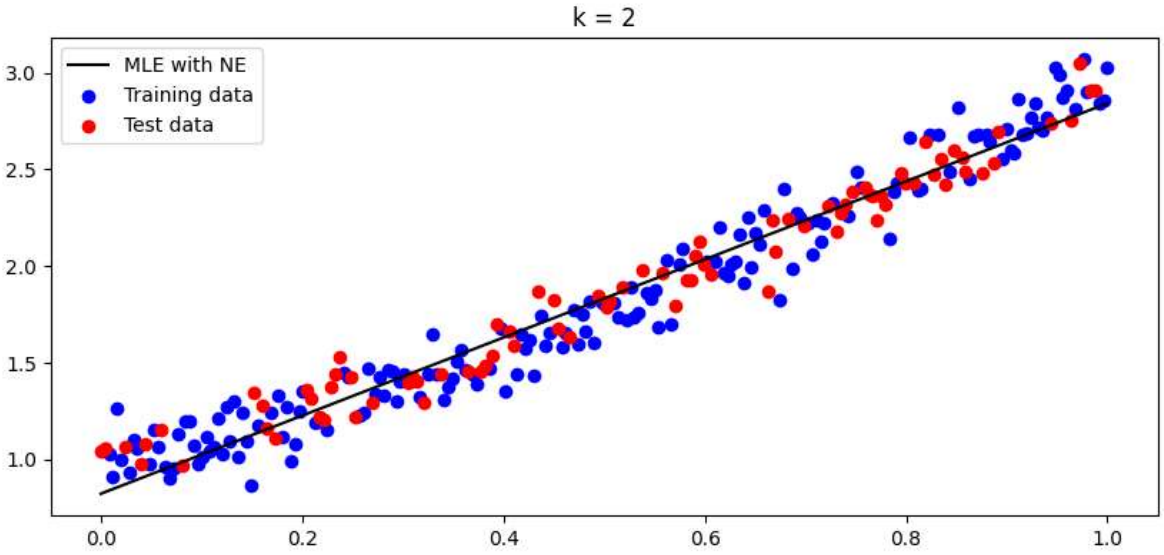

k = 5
0.16
0.14
0.12
0.10
0.08
−0.04
−0.02
0.00
0.02
0.04
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10

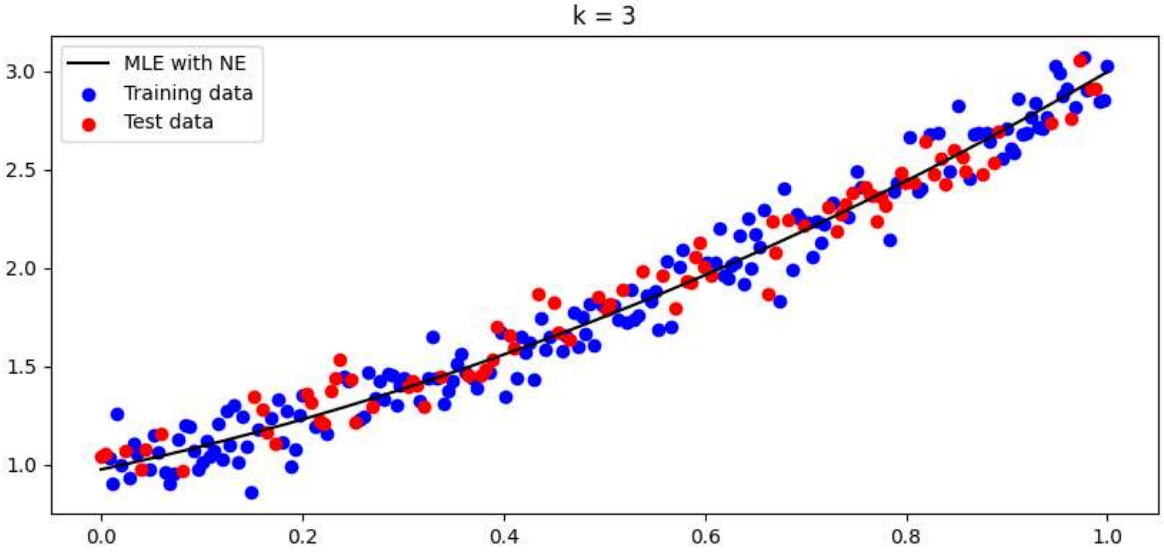

k = 6
0.12
0.11
0.10
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10

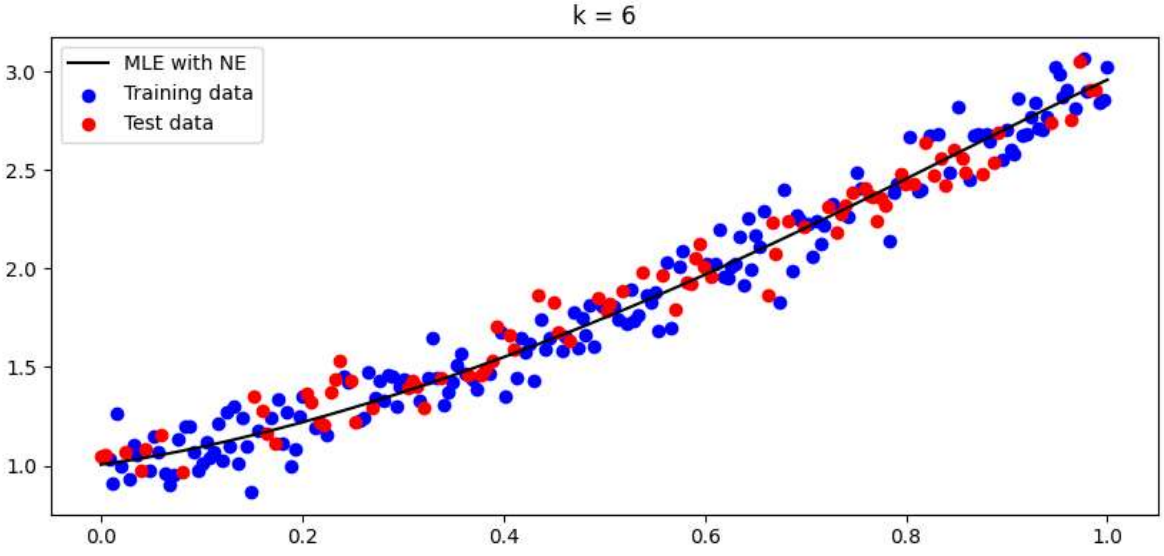


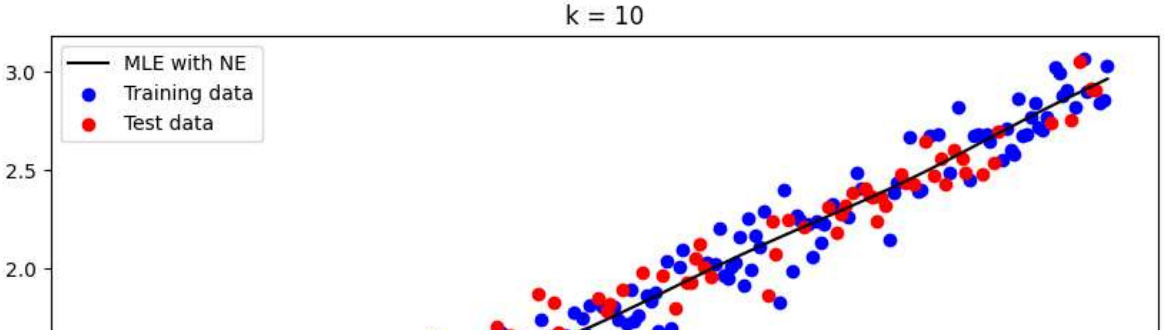
N = 250

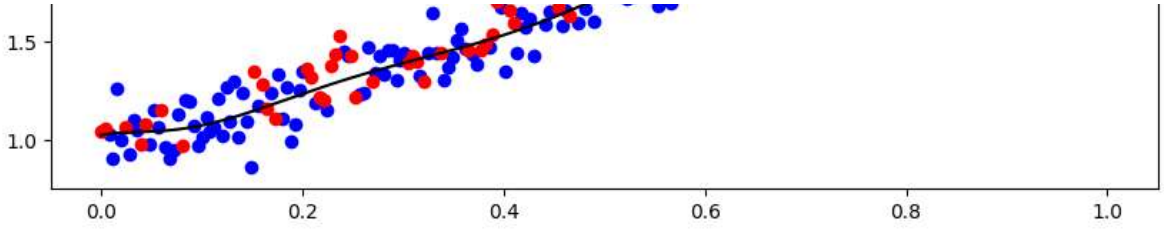
Method: NE

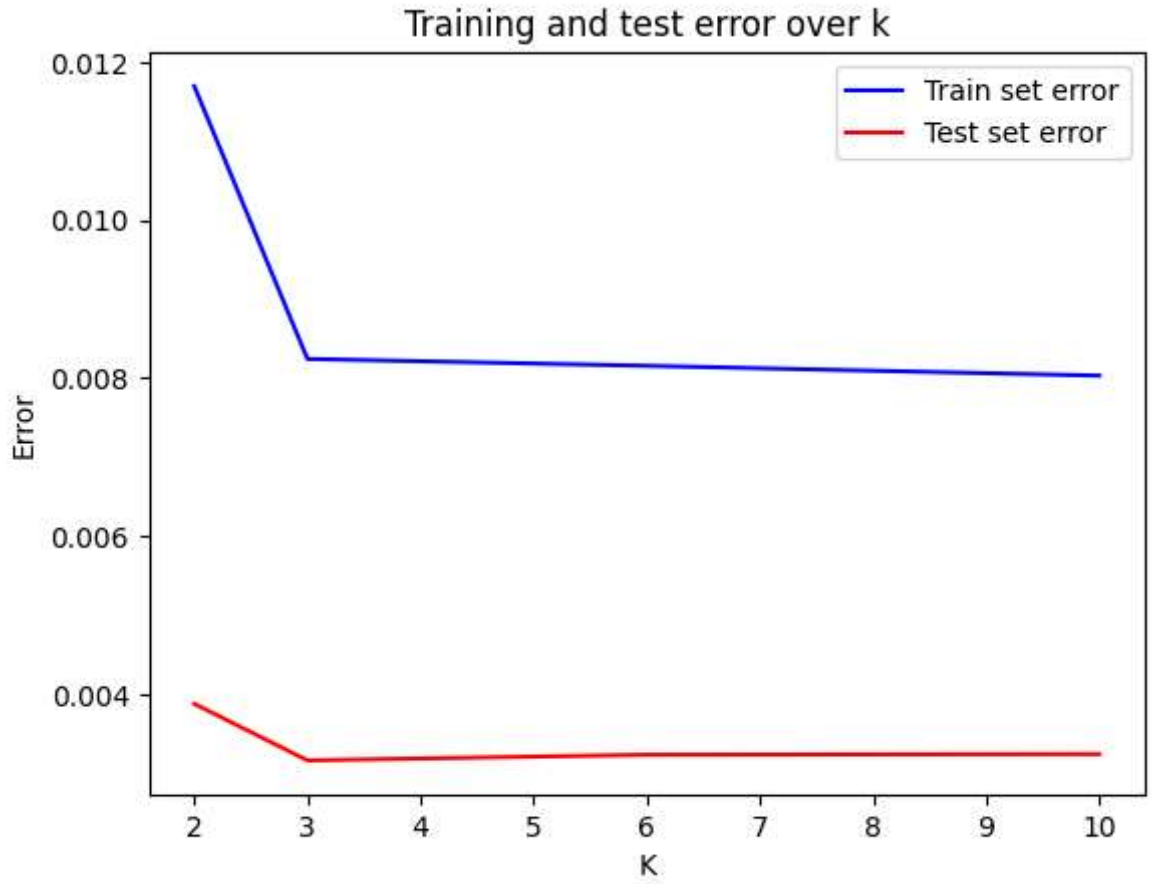

k = 2
MLE with NE
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0


k = 3
MLE with NE
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0

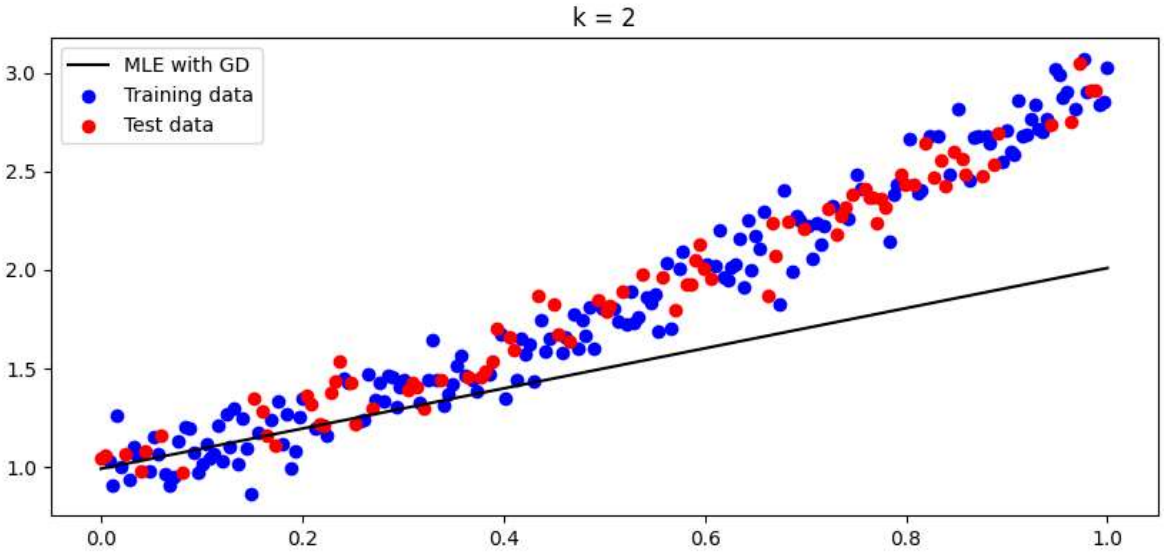

k = 6
MLE with NE
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0

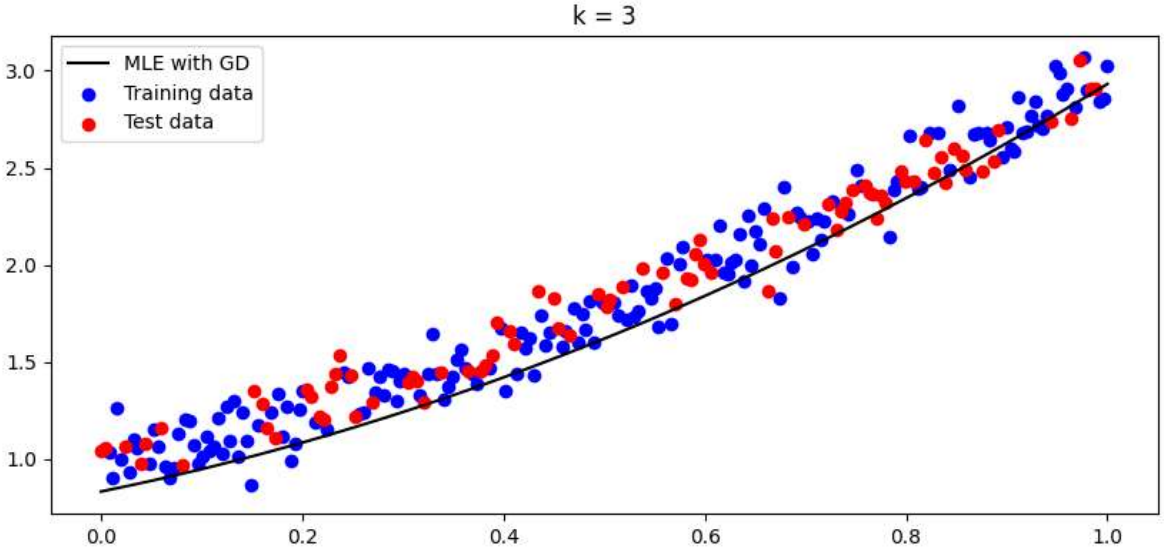

k = 10
MLE with NE
Training data
Test data
3.0
2.5
2.0

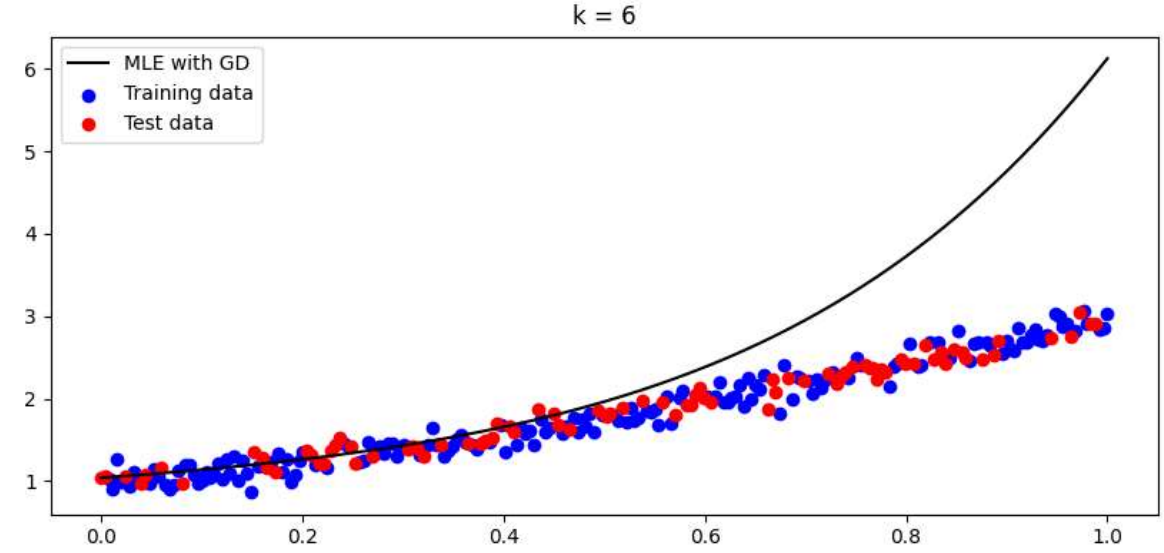

1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0

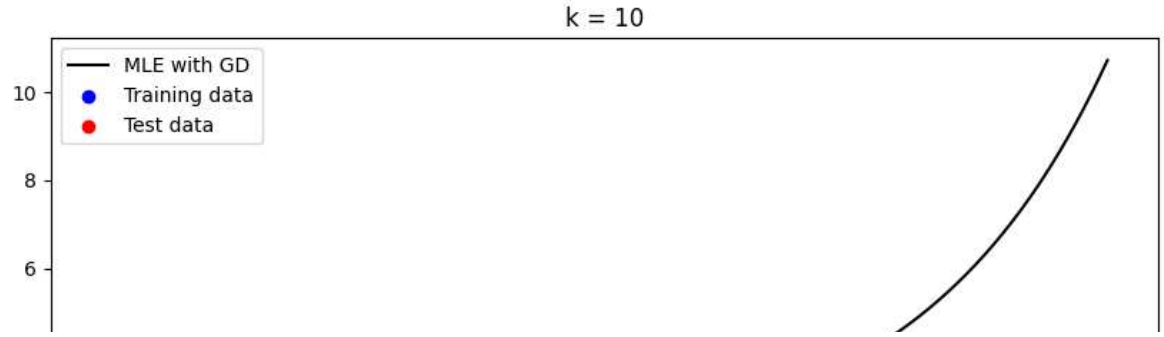

Training and test error over k
Train set error
Test set error
Error
0.012
0.010
0.008
0.006
0.004
2
3
4
5
6
7
8
9
10
K

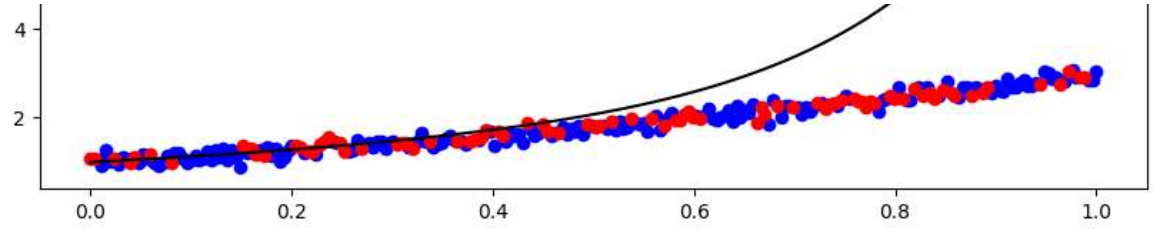
Method: GD


k = 2
MLE with GD
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0


k = 3
MLE with GD
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0

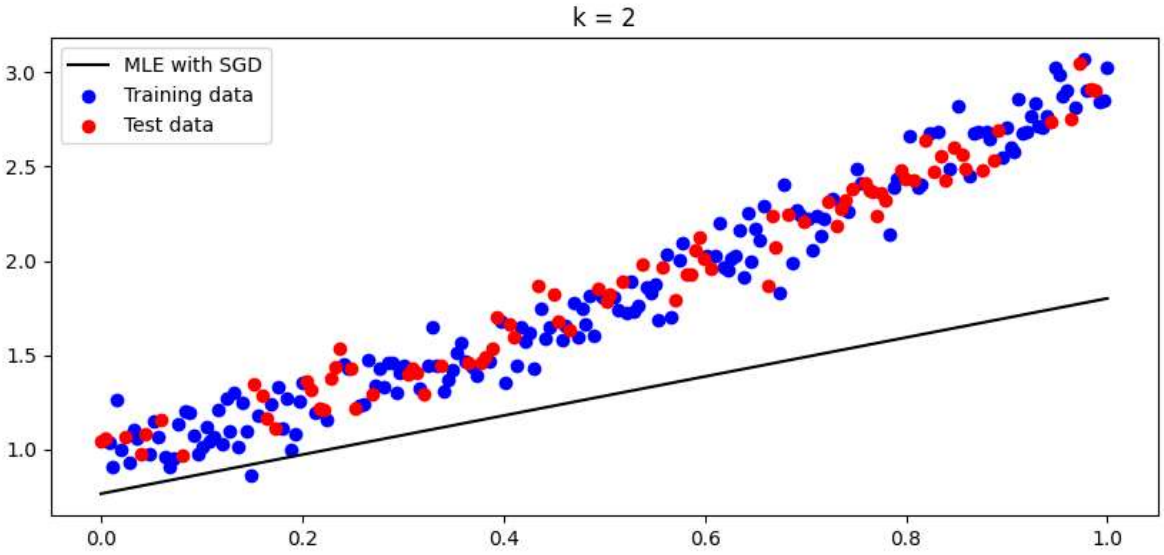

k = 6
MLE with GD
Training data
Test data
6
5
4
3
2
1
0.0
0.2
0.4
0.6
0.8
1.0

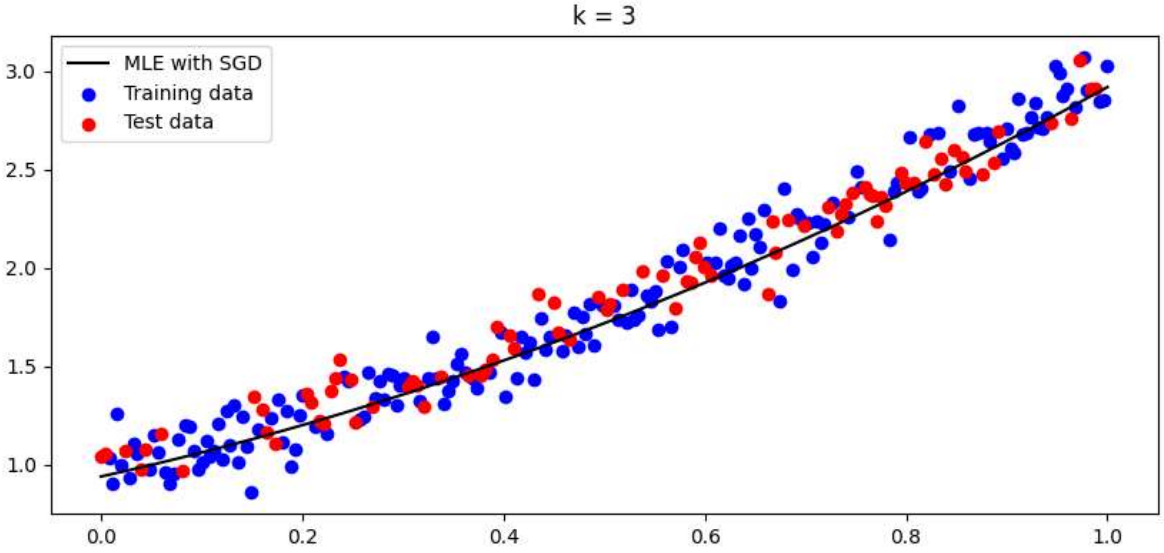

k = 10
MLE with GD
Training data
Test data
10
8
6

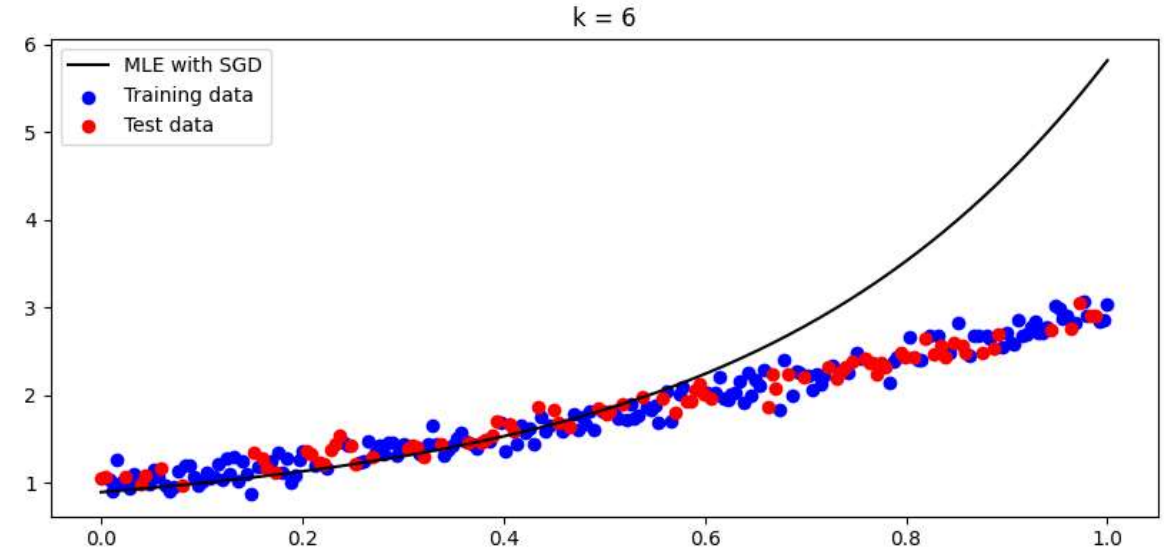

4
2
0.0
0.2
0.4
0.6
0.8
1.0

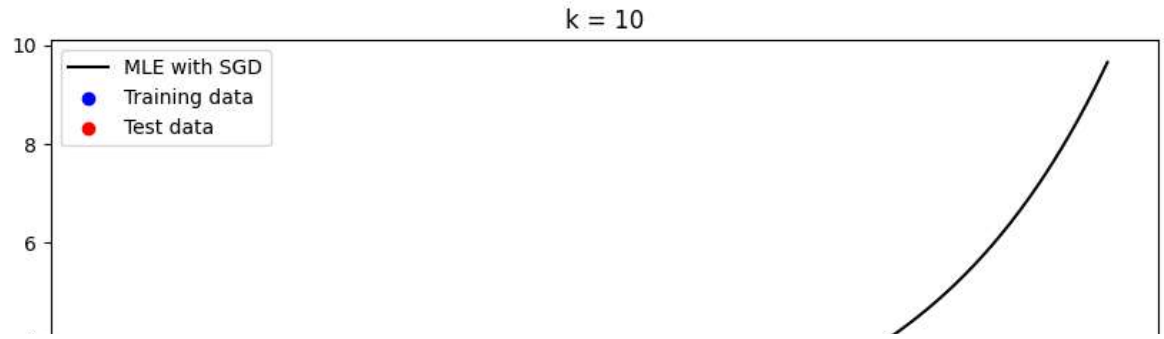

Training and test error over k
Train set error
Test set error
Error
K
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2
3
4
5
6
7
8
9
10

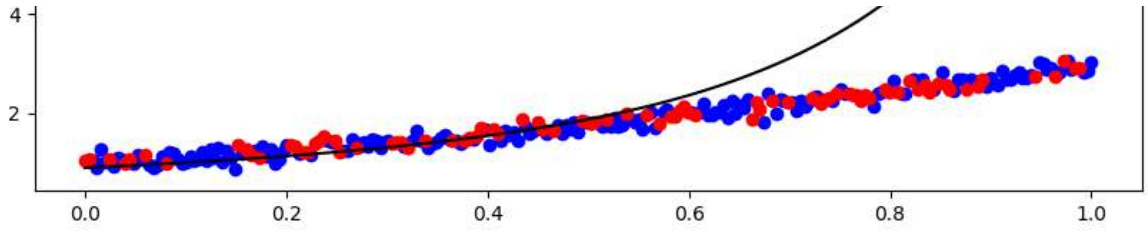
```
#### Method: SGD ####
```


k = 2
MLE with SGD
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0


k = 3
MLE with SGD
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0

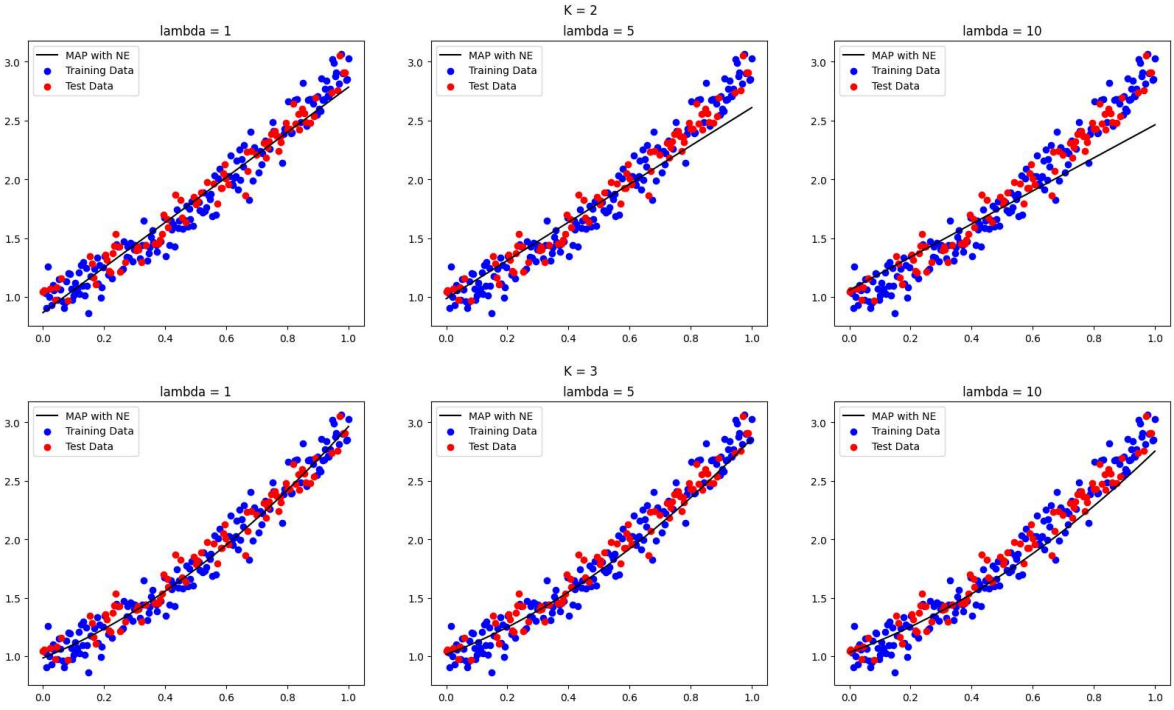

k = 6
MLE with SGD
Training data
Test data
6
5
4
3
2
1
0.0
0.2
0.4
0.6
0.8
1.0


k = 10
MLE with SGD
Training data
Test data
10
8
6

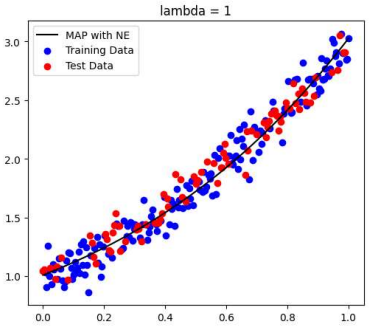

4
2
0.0
0.2
0.4
0.6
0.8
1.0

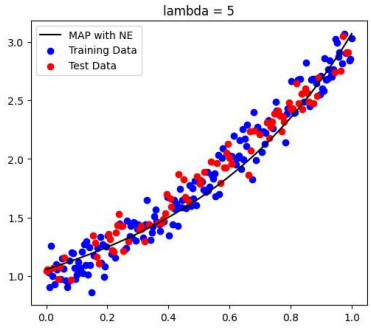

Training and test error over k
Train set error
Test set error
Error
K
2.5
2.0
1.5
1.0
0.5
0.0
2
3
4
5
6
7
8
9
10

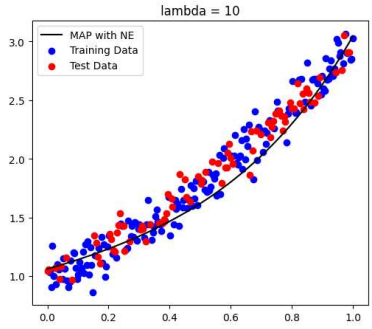
Method: NE


K = 2
lambda = 1
lambda = 5
lambda = 10
K = 3
lambda = 1
lambda = 5
lambda = 10
MAP with NE
Training Data
Test Data

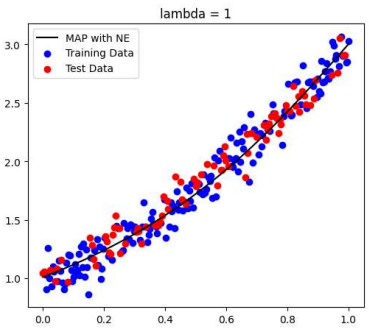
K = 6

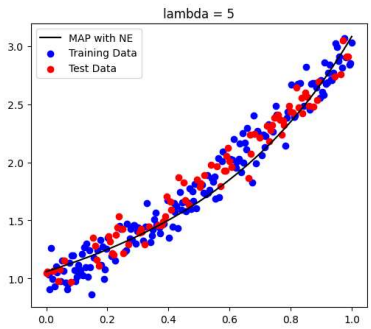


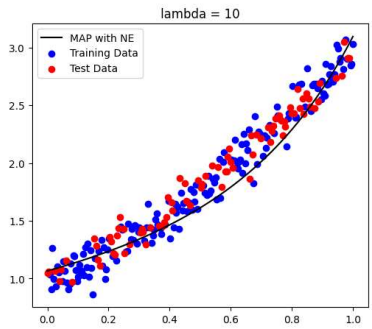




K = 8

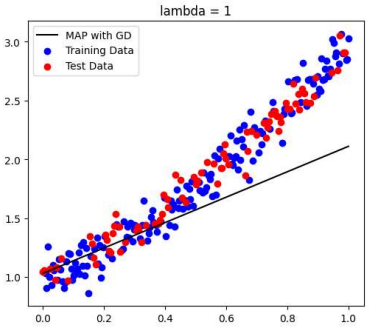


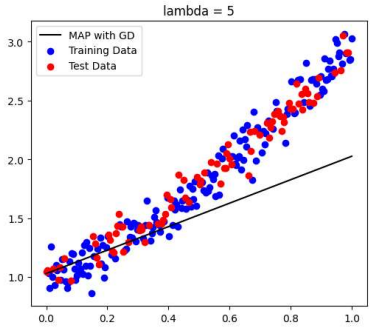


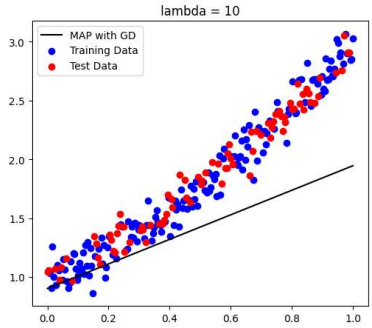


Method: GD

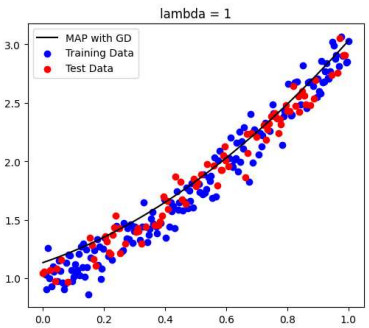
K = 2

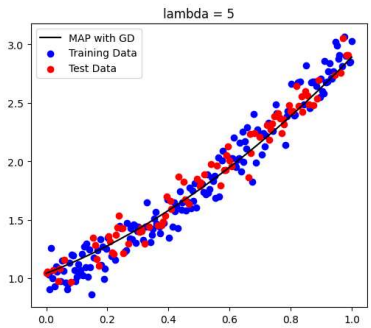


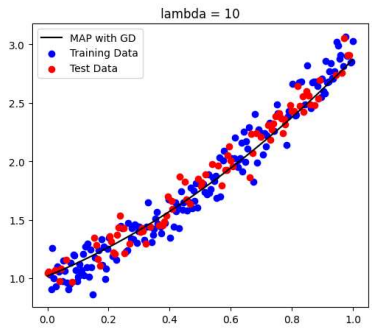




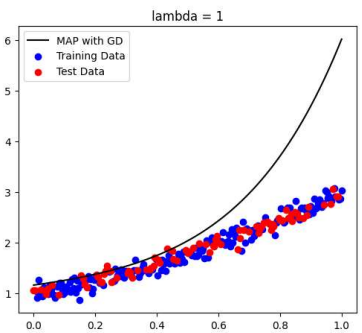
K = 3

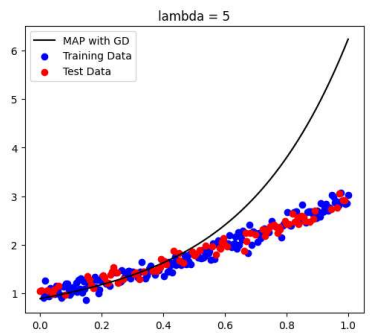


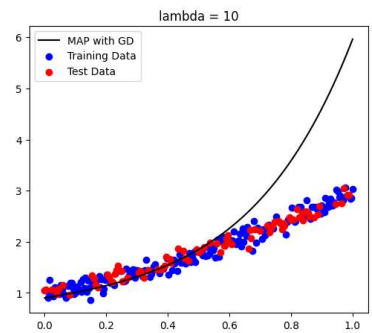




K = 6

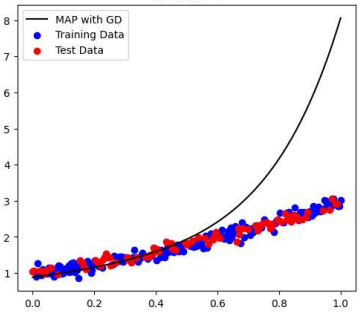




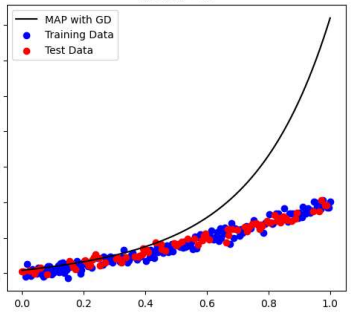


K = 8

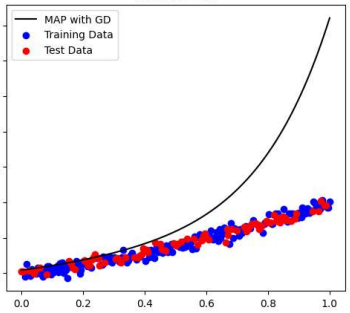
lambda = 1



lambda = 5



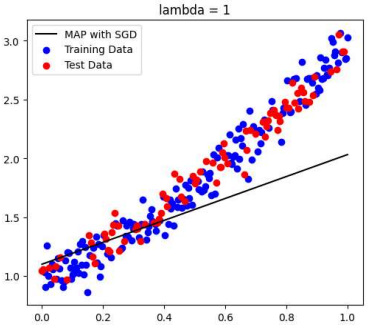
lambda = 10



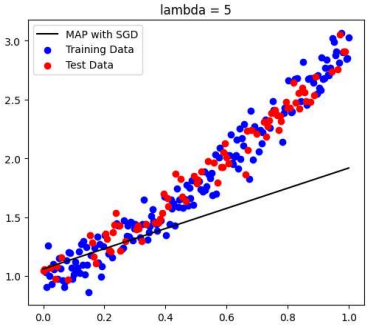
Method: SGD

K = 2

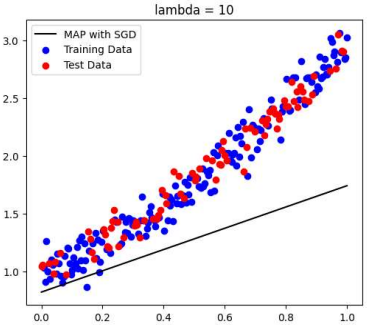
lambda = 1



lambda = 5

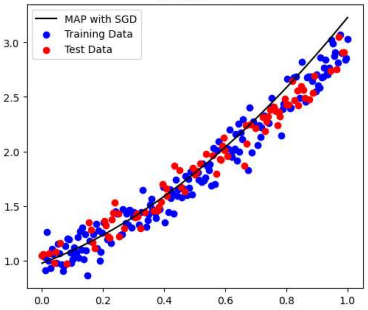


lambda = 10

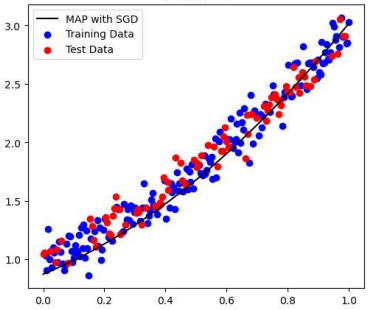


K = 3

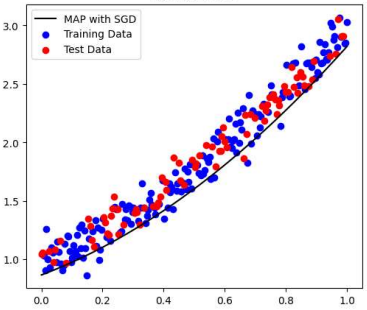
lambda = 1



lambda = 5

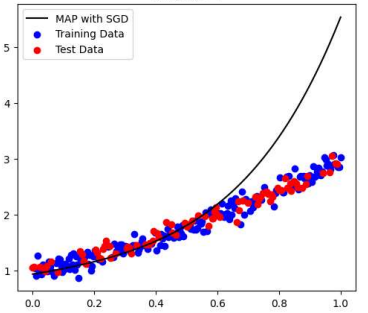


lambda = 10

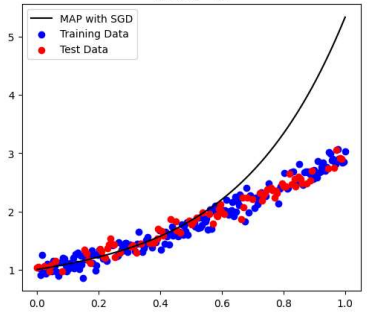


K = 6

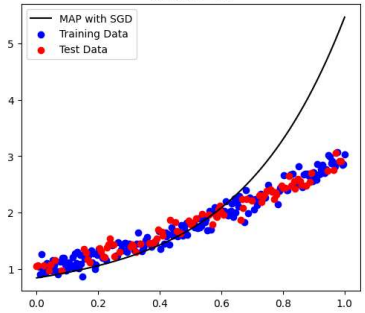
lambda = 1



lambda = 5

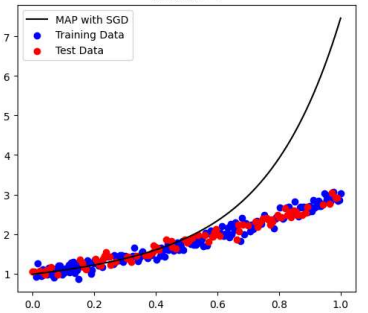


lambda = 10

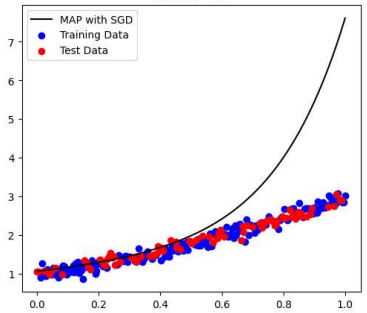


K = 8

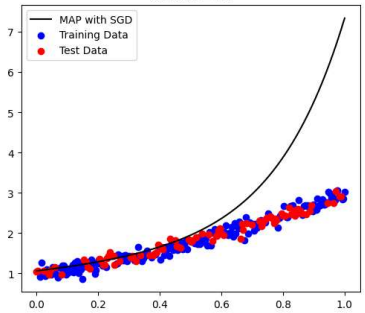
lambda = 1



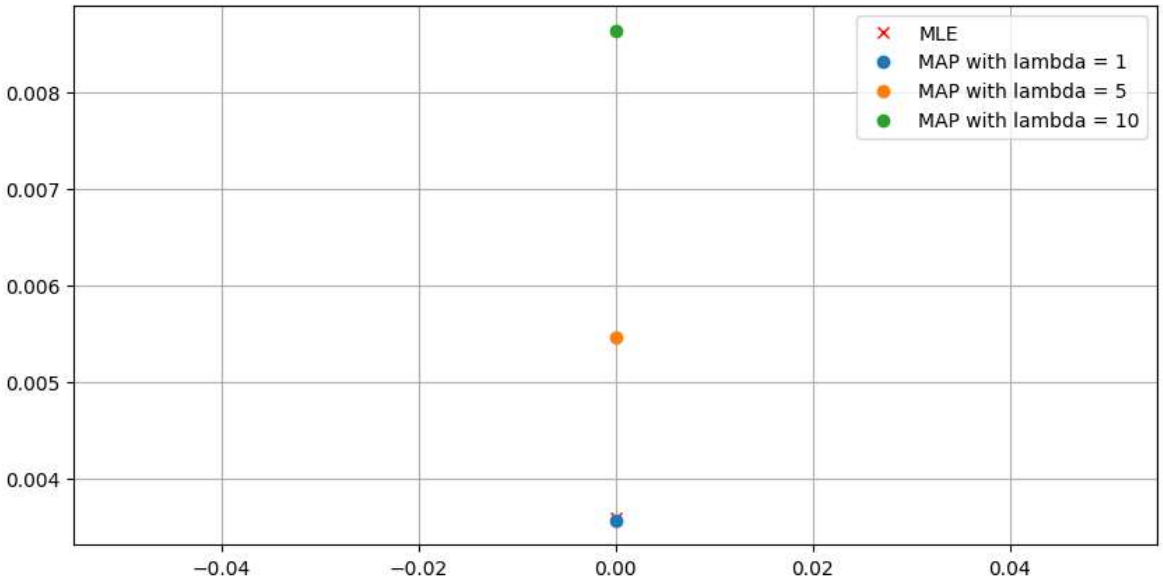
lambda = 5

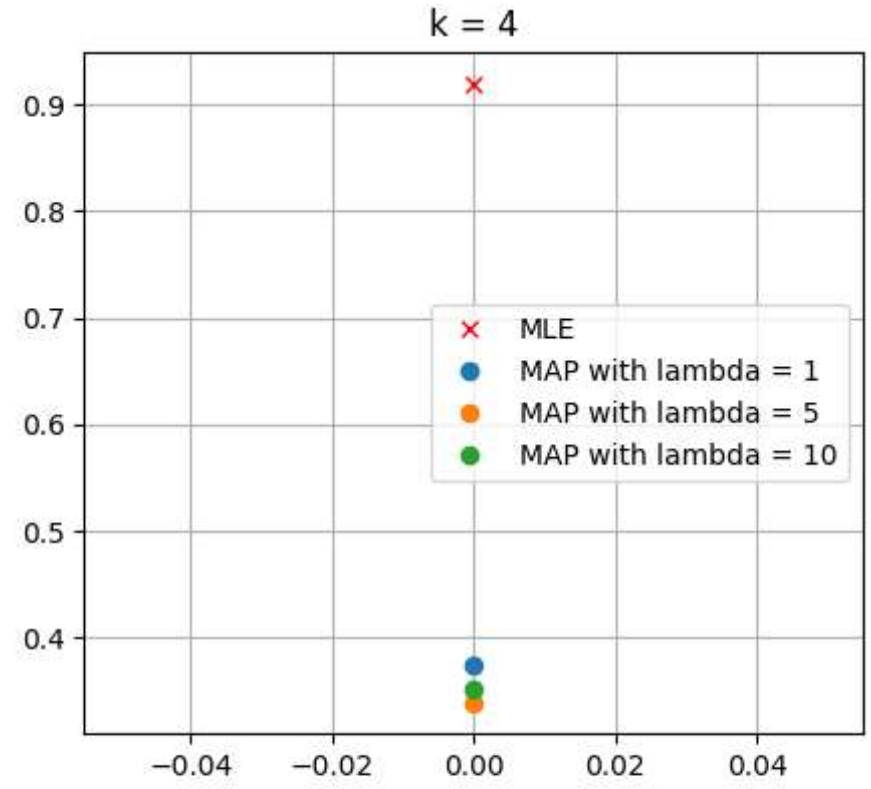


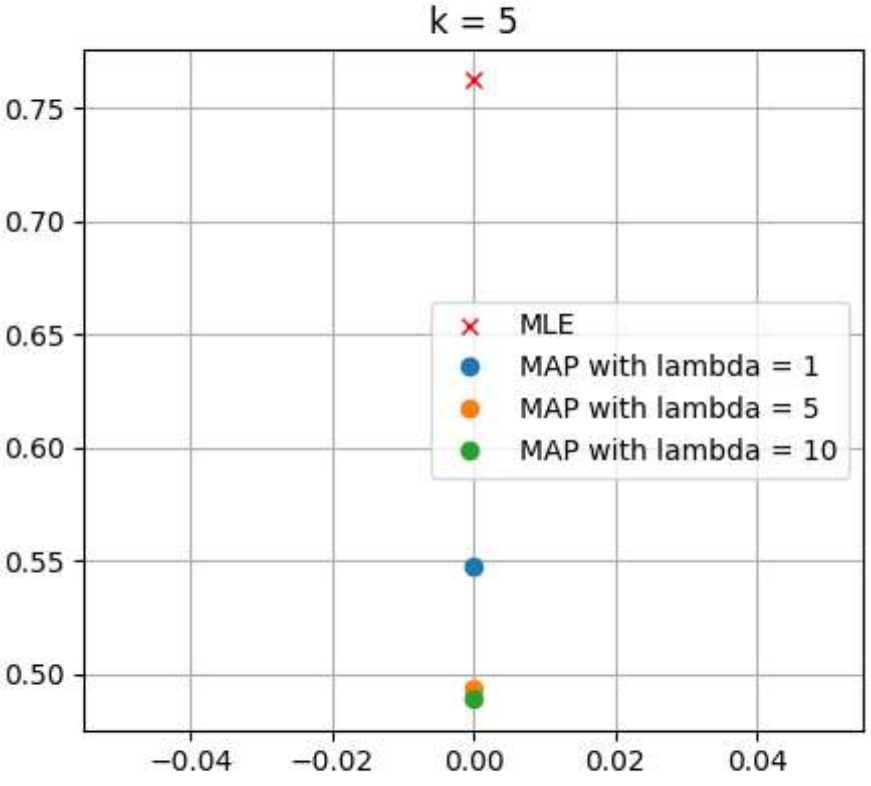
lambda = 10

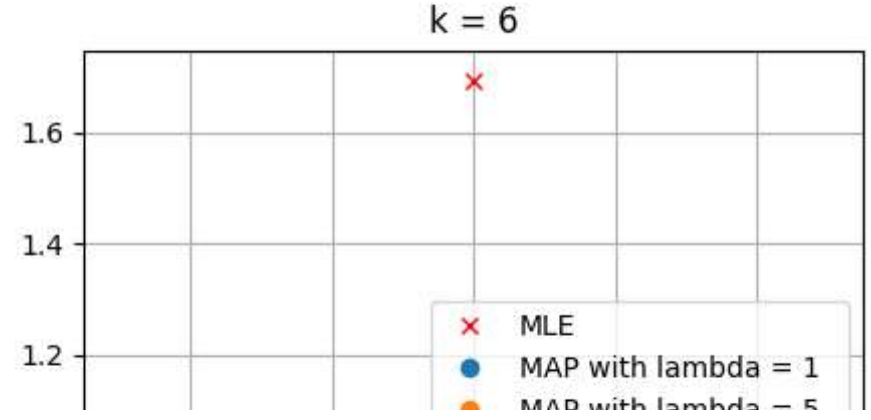


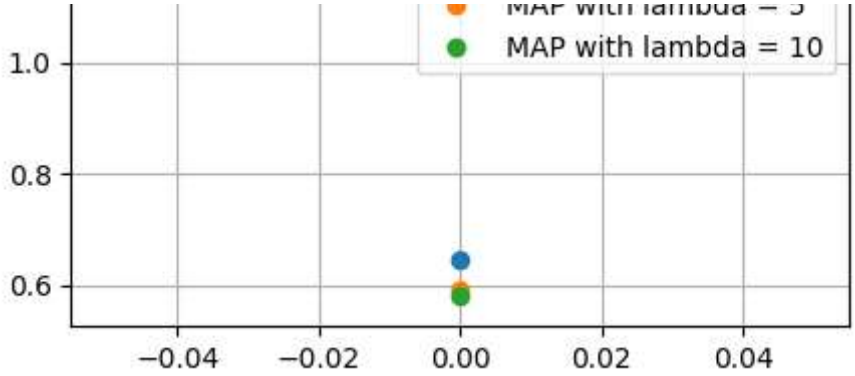
```
Theta MLE GD: [0.94394792 1.10989891 1.21137023]
Error MLE GD: 0.1412998957966383
Theta MLE SGD: [1.03190999 1.01749107 0.88677066]
Error MLE SGD: 0.1412998957966383
Theta MLE NE: [0.97597165 1.08265702 0.93583021]
Error MLE NE: 0.06198738292796218

Theta MAP GD: [1.03965828 1.04532139 1.06289964]
Error MAP GD: 0.05027647667073202
Theta MAP SGD: [0.87856212 0.92558242 0.91968417]
Error MAP SGD: 0.05027647667073202
Theta MAP NE: [0.98664816 1.0605299  0.91789044]
Error MAP NE: 0.059397306113444424


Method: NE


MLE test error:  0.003593703114384349
MAP test error:  [0.0035691887524784, 0.0054633757974872224, 0.00864401204287205
1]
```




k = 4
0.9
0.8
0.7
0.6
0.5
0.4
−0.04
−0.02
0.00
0.02
0.04
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10

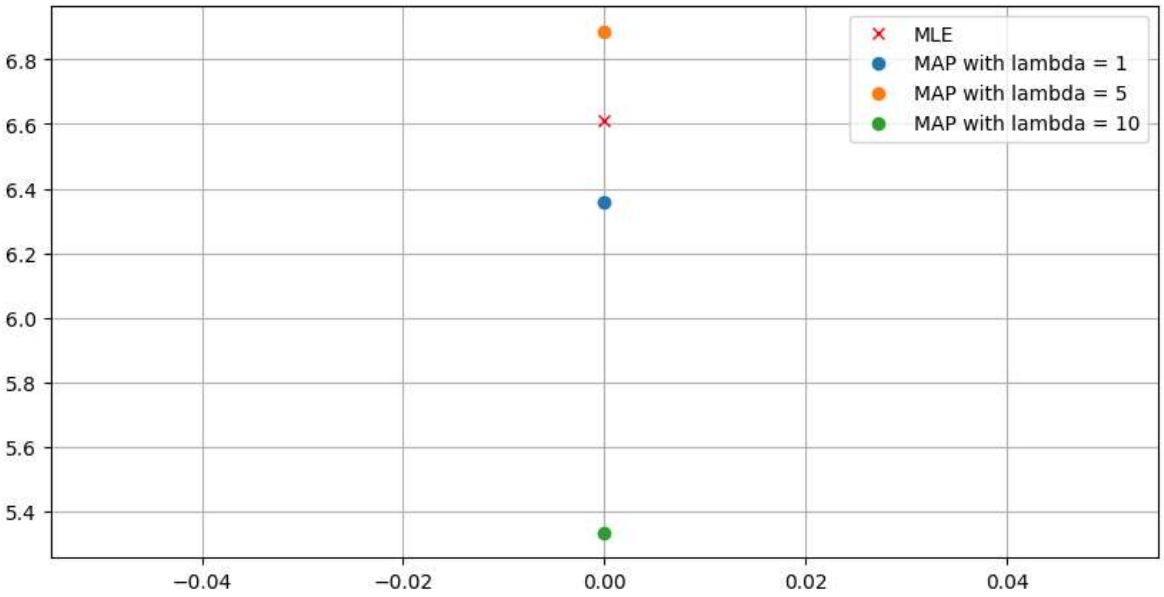

k = 5
0.75
0.70
0.65
0.60
0.55
0.50
−0.04
−0.02
0.00
0.02
0.04
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10

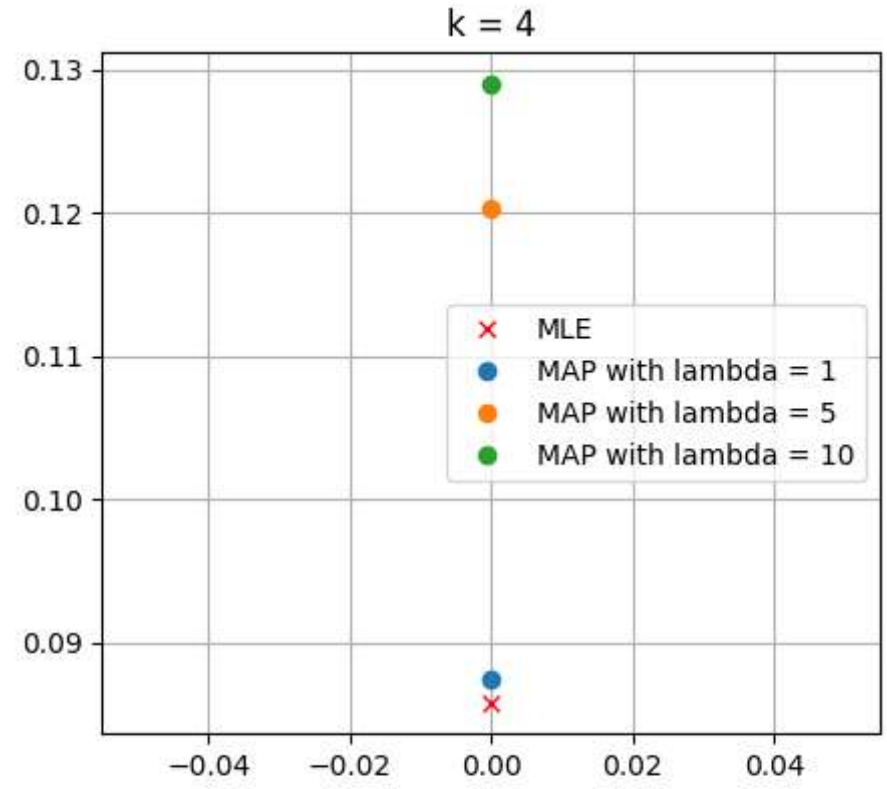

k = 6
1.6
1.4
1.2
MLE
MAP with lambda = 1

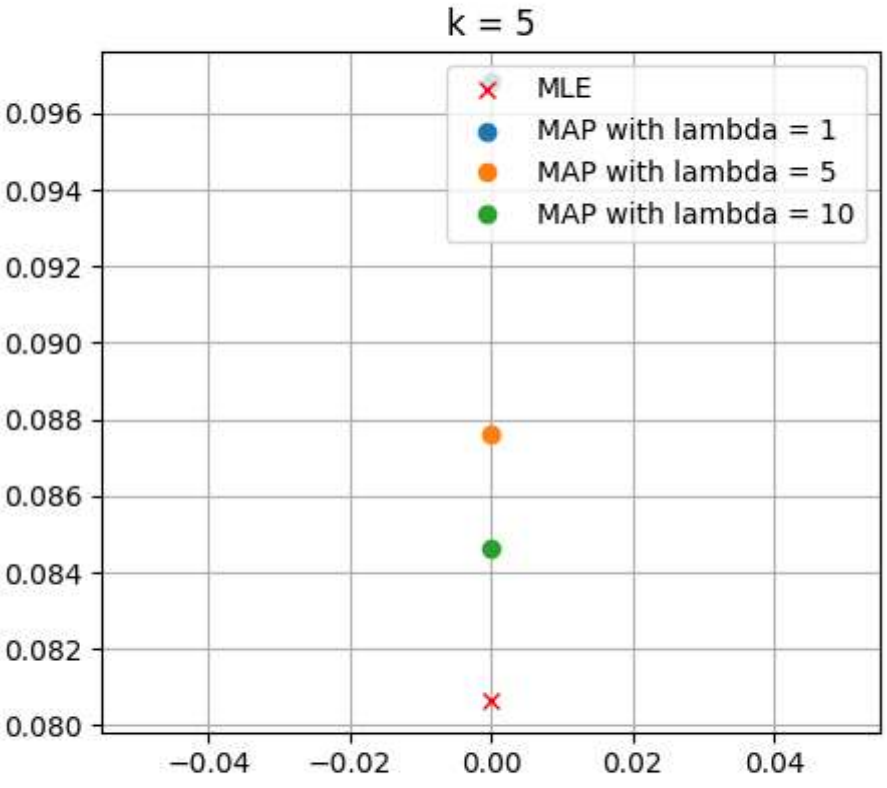


Method: GD

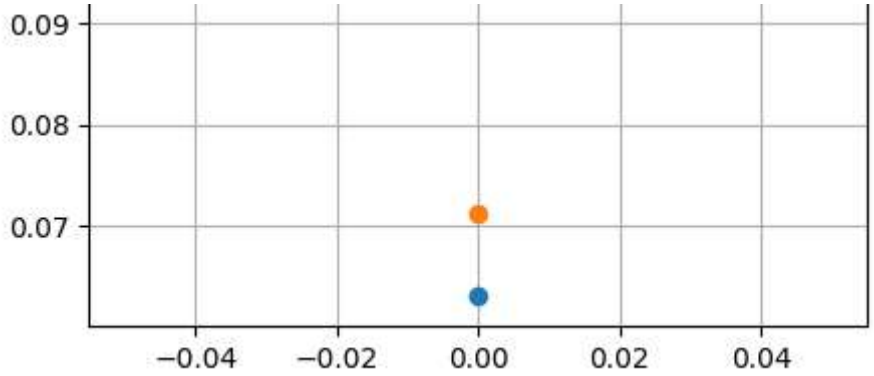
MLE test error: 6.610802206729946
MAP test error: [6.359299985757039, 6.886814994756625, 5.334981143643618]




k = 4
0.13
0.12
0.11
0.10
0.09
−0.04
−0.02
0.00
0.02
0.04
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10

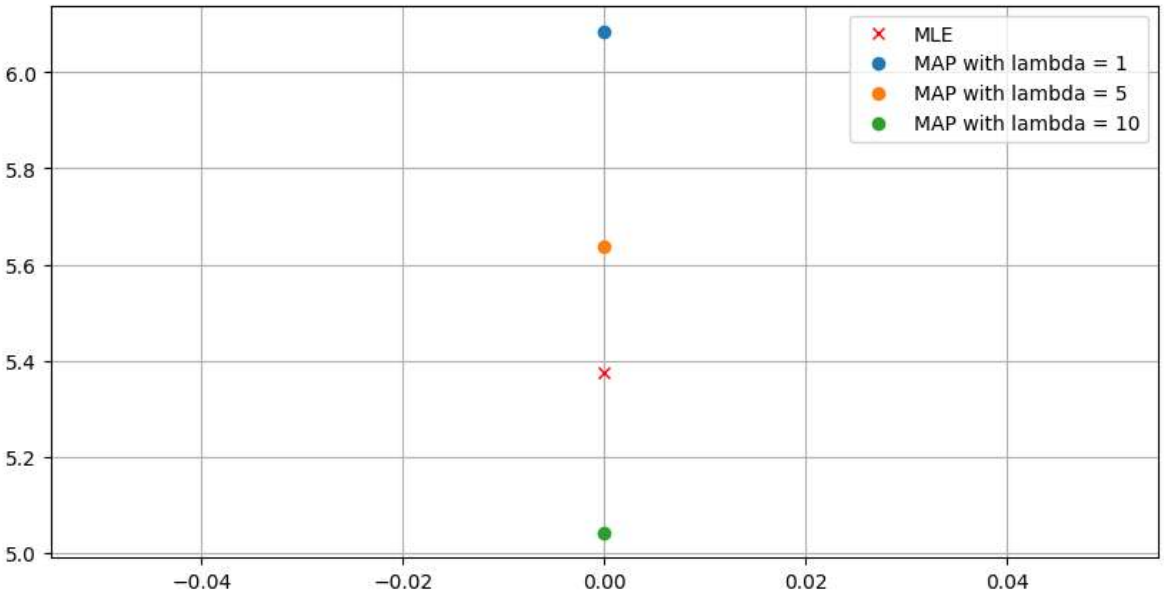

k = 5
0.096
0.094
0.092
0.090
0.088
0.086
0.084
0.082
0.080
−0.04
−0.02
0.00
0.02
0.04
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10

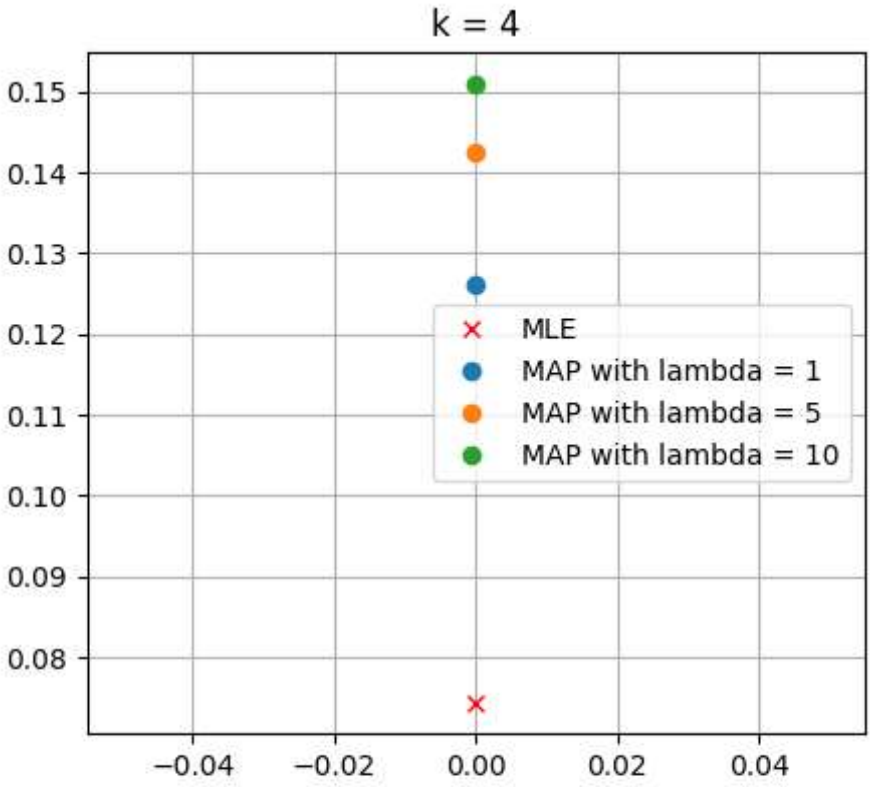

k = 6
0.12
0.11
0.10
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10

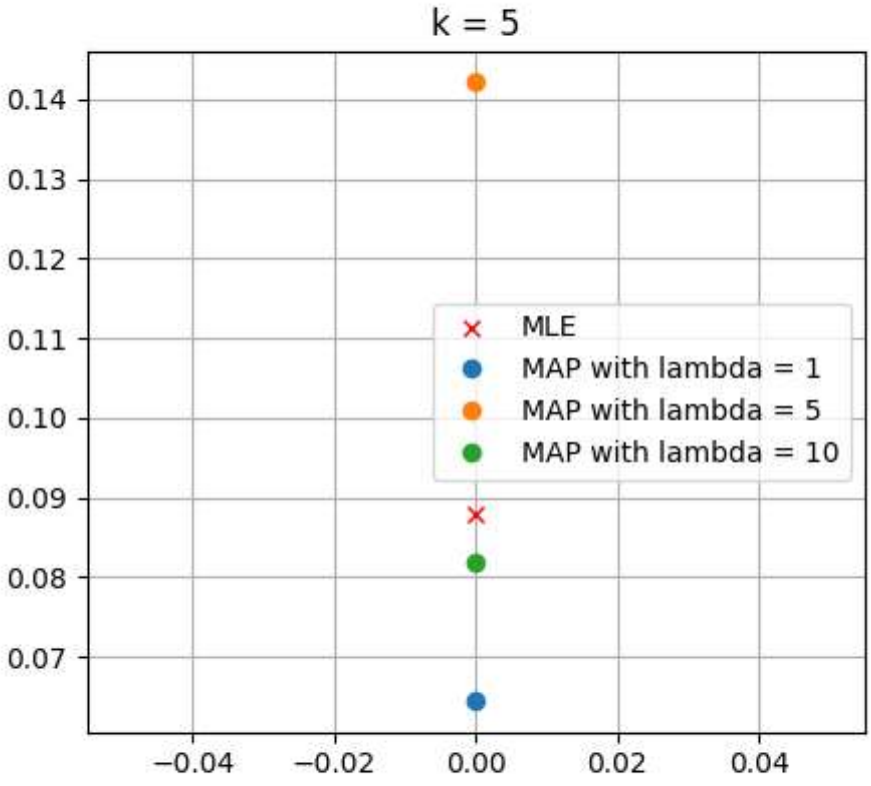


Method: SGD

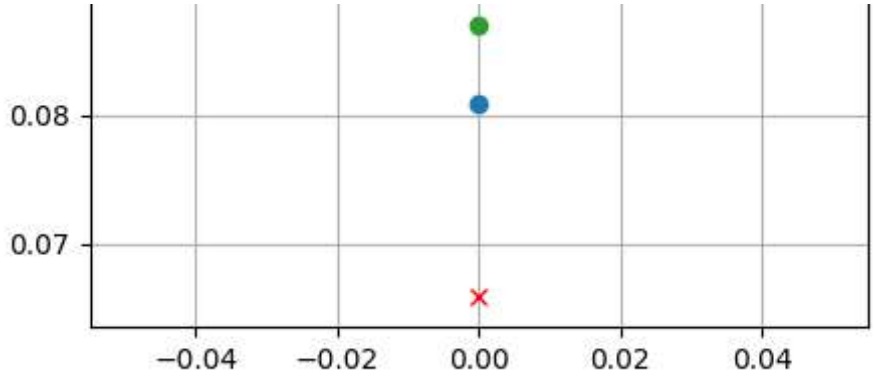
MLE test error: 5.375338784107616
MAP test error: [6.08507405150407, 5.63761456739814, 5.042181529127683]




k = 4
0.15
0.14
0.13
0.12
0.11
0.10
0.09
0.08
−0.04
−0.02
0.00
0.02
0.04
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10

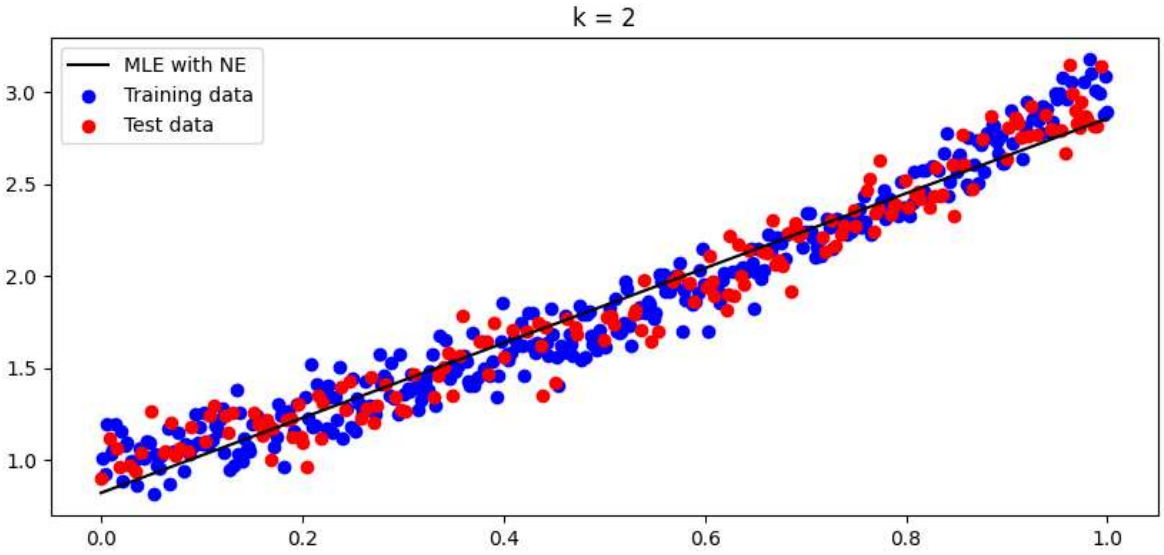

k = 5
0.14
0.13
0.12
0.11
0.10
0.09
0.08
0.07
−0.04
−0.02
0.00
0.02
0.04
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10

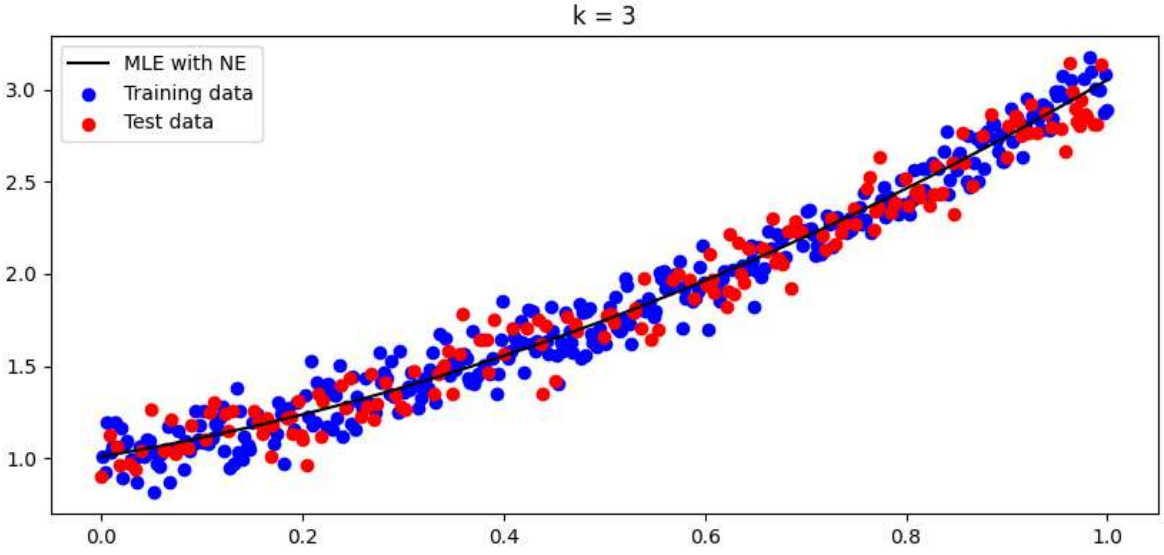

k = 6
0.11
0.10
0.09
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10

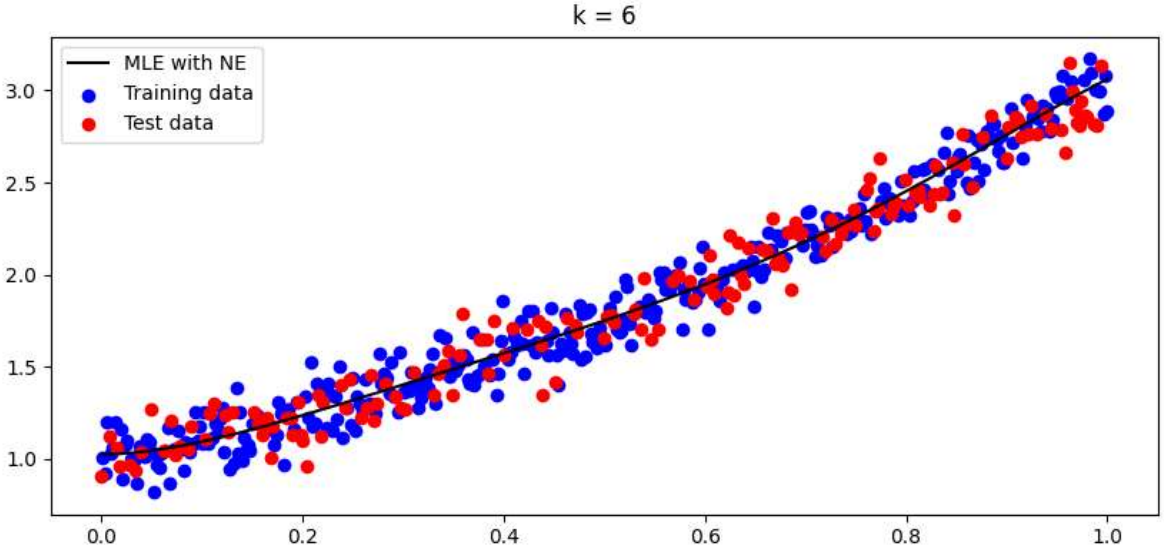


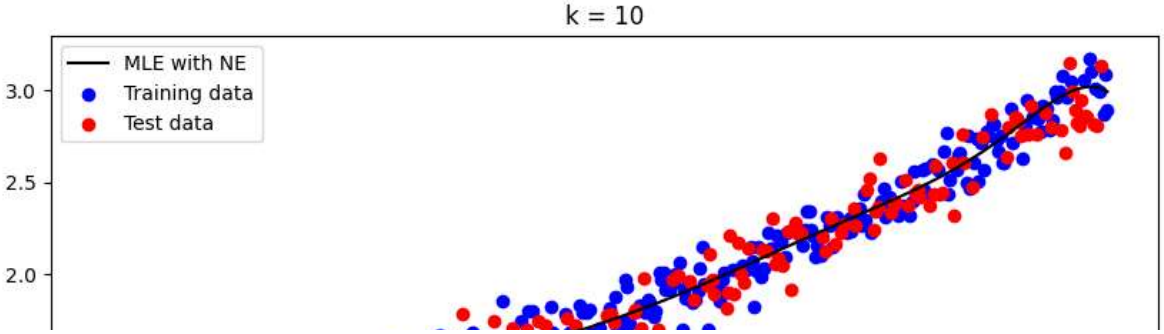
N = 500

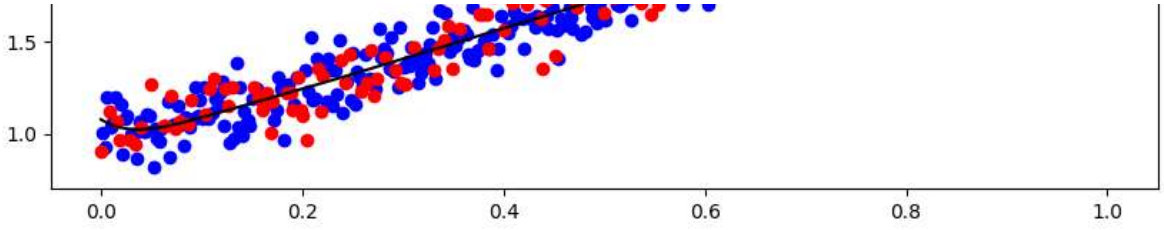
Method: NE


k = 2
MLE with NE
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0


k = 3
MLE with NE
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0

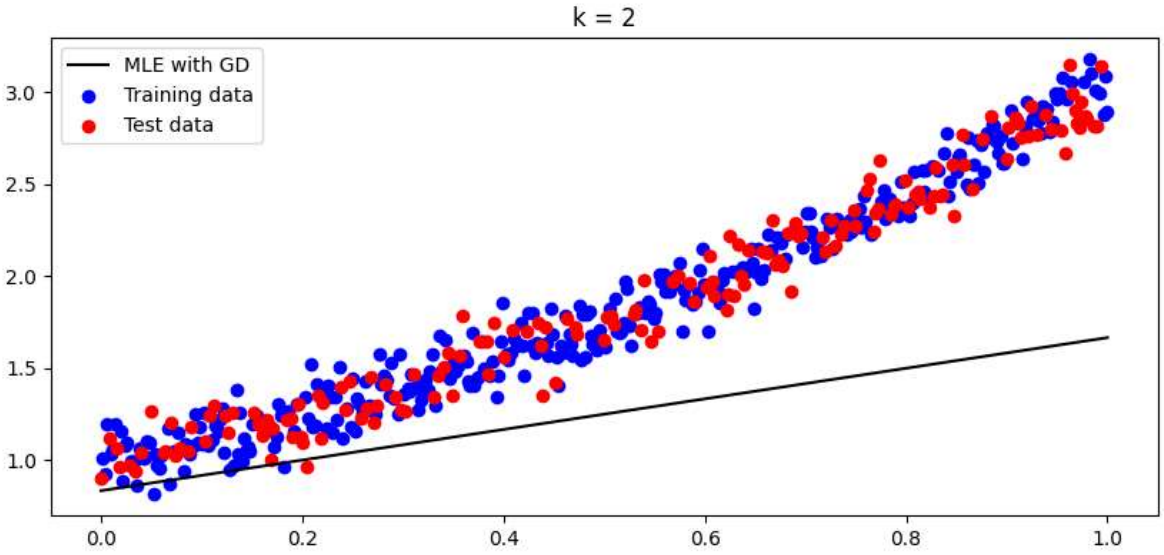

k = 6
MLE with NE
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0

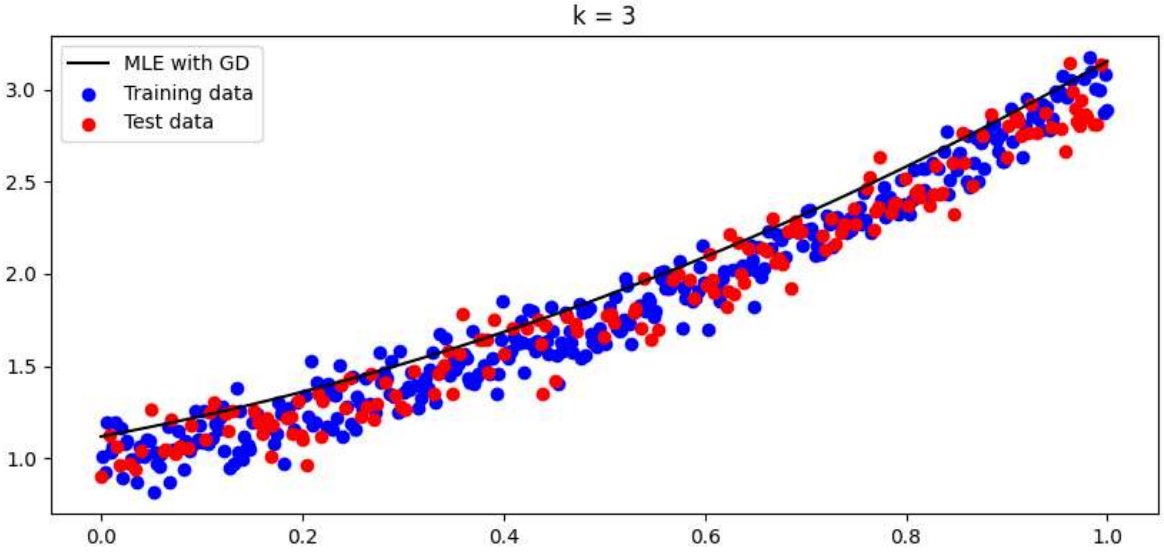

k = 10
MLE with NE
Training data
Test data
3.0
2.5
2.0

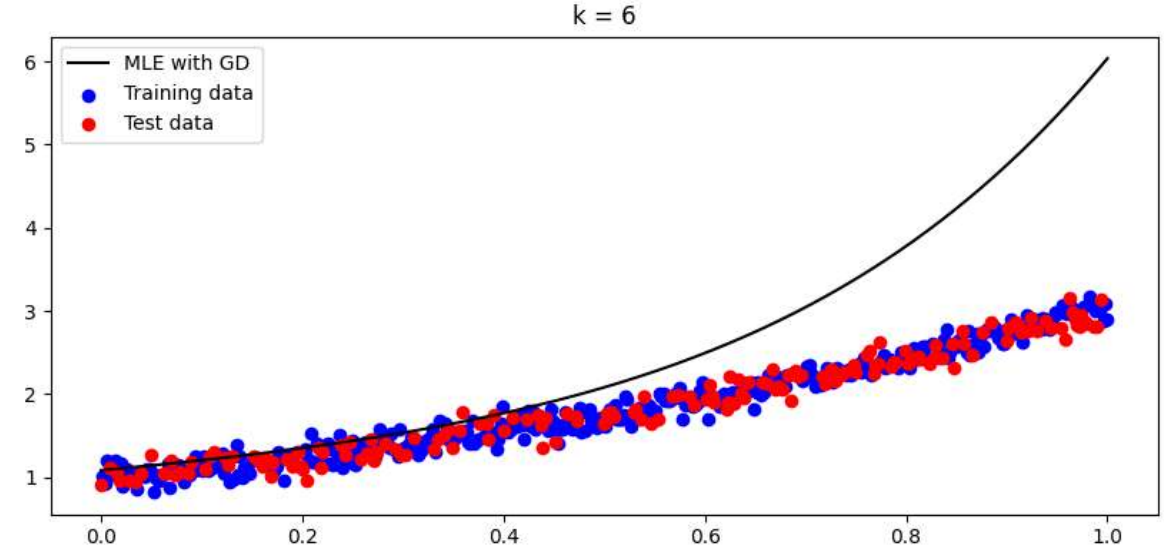

1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0

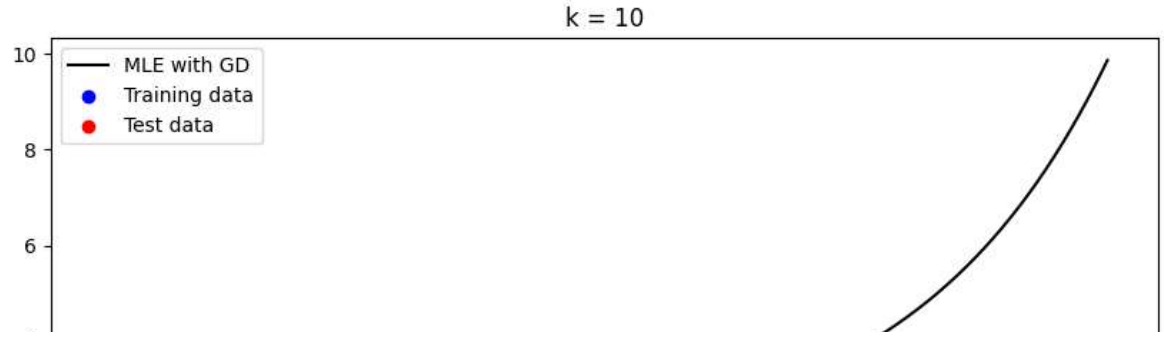

Training and test error over k
Train set error
Test set error
Error
K
0.012
0.011
0.010
0.009
0.008
0.007
0.006
0.005
0.004
2
3
4
5
6
7
8
9
10

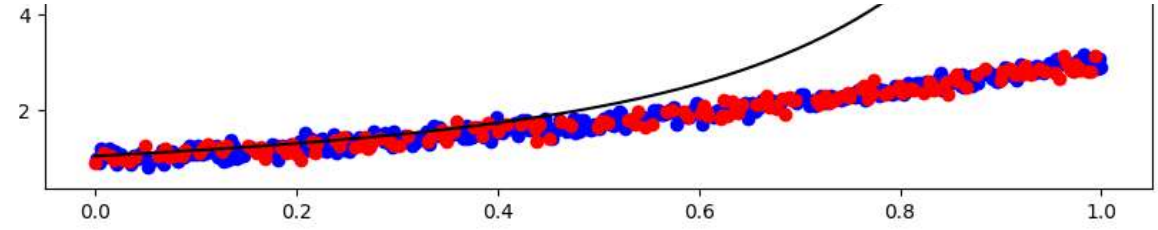
```
#### Method: GD ####
```


k = 2
MLE with GD
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0


k = 3
MLE with GD
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0

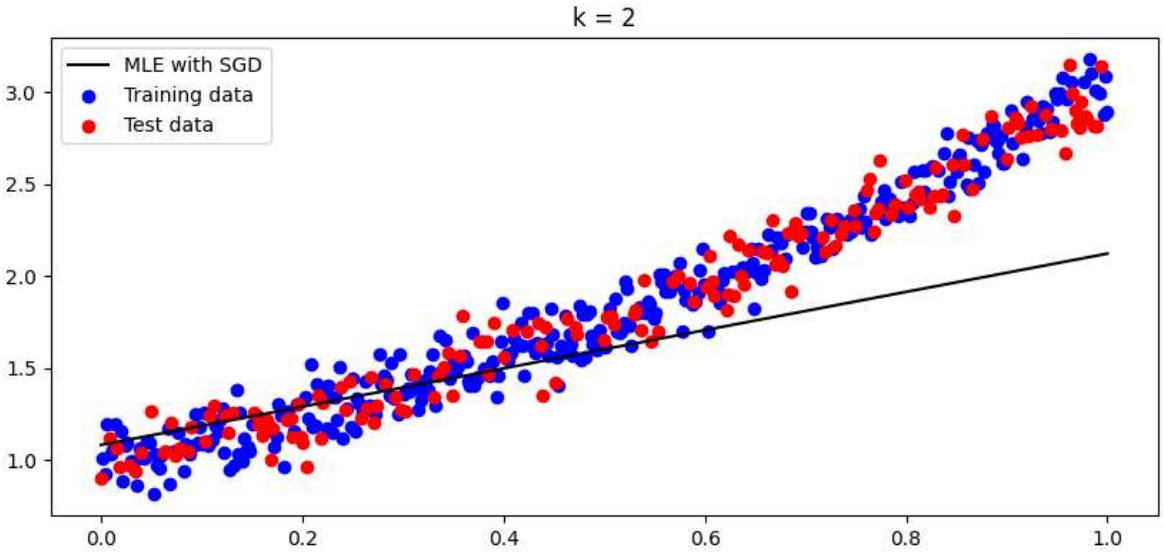

k = 6
MLE with GD
Training data
Test data
6
5
4
3
2
1
0.0
0.2
0.4
0.6
0.8
1.0

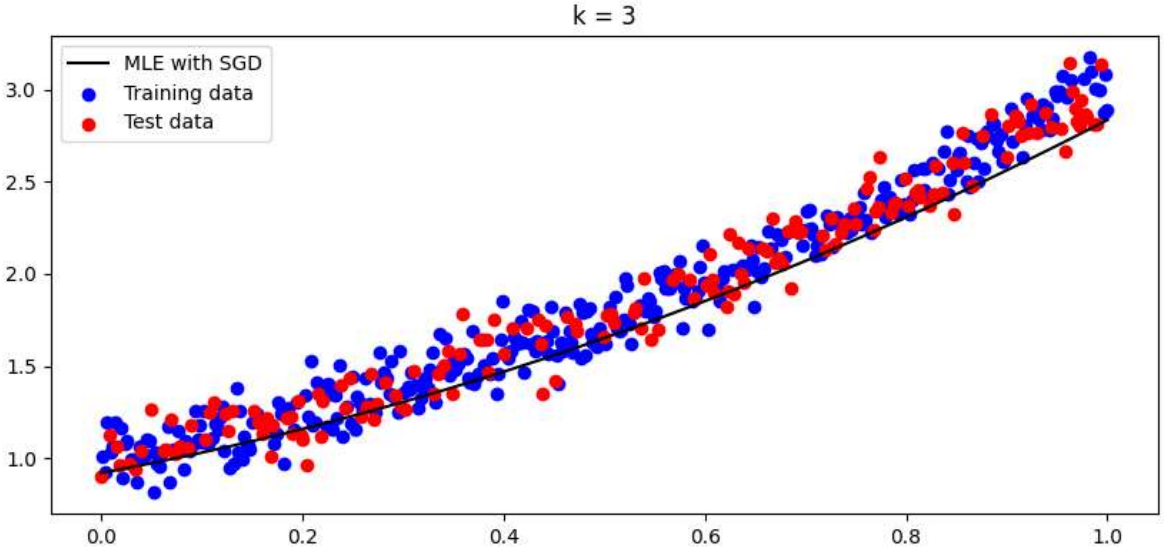

k = 10
MLE with GD
Training data
Test data
10
8
6

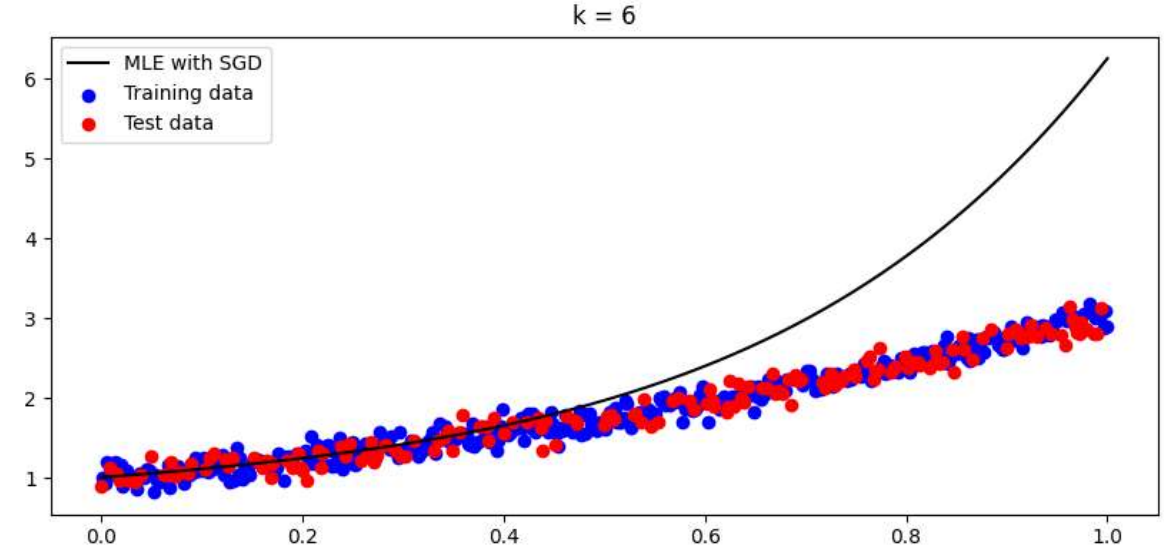

4
2
0.0
0.2
0.4
0.6
0.8
1.0

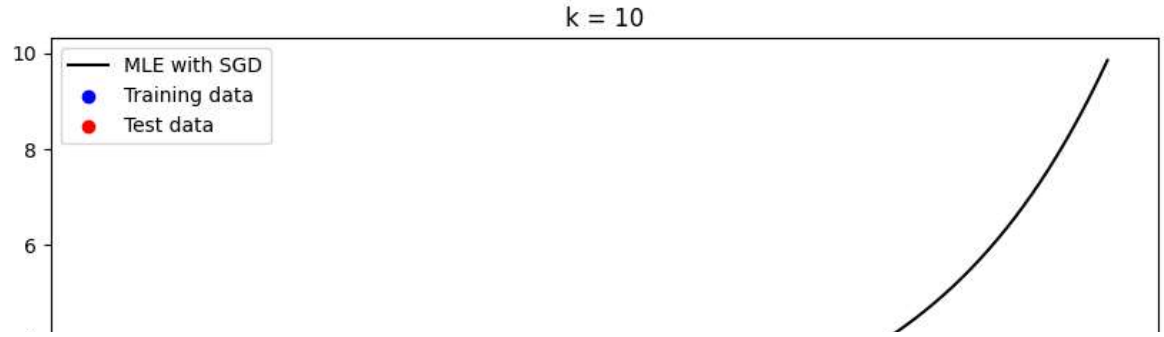

Training and test error over k
Train set error
Test set error
Error
K
2.0
1.5
1.0
0.5
0.0
2
3
4
5
6
7
8
9
10

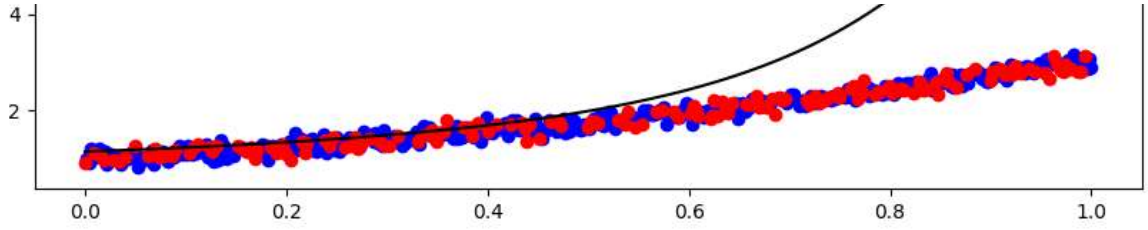
Method: SGD


k = 2
MLE with SGD
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0


k = 3
MLE with SGD
Training data
Test data
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0

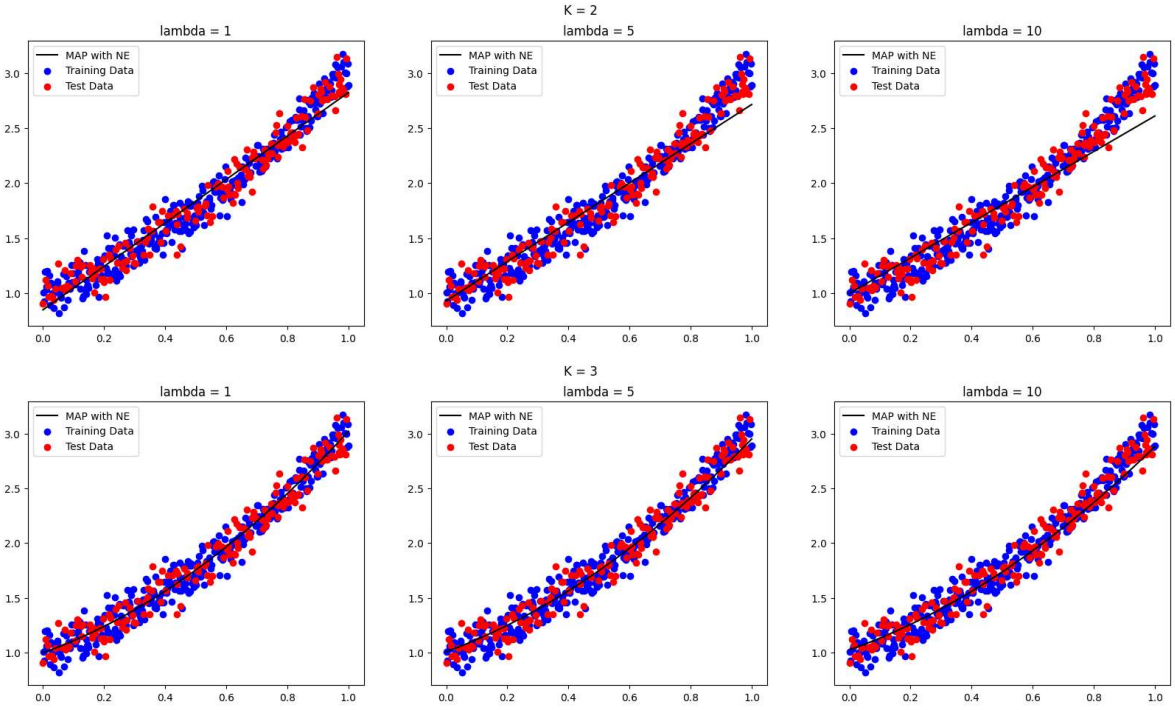

k = 6
MLE with SGD
Training data
Test data
6
5
4
3
2
1
0.0
0.2
0.4
0.6
0.8
1.0


k = 10
MLE with SGD
Training data
Test data
10
8
6

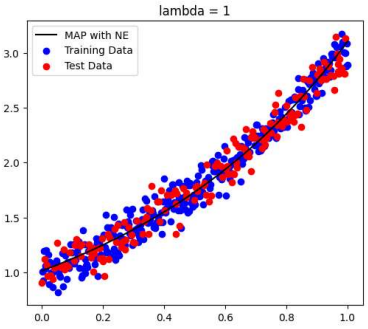

4
2
0.0
0.2
0.4
0.6
0.8
1.0

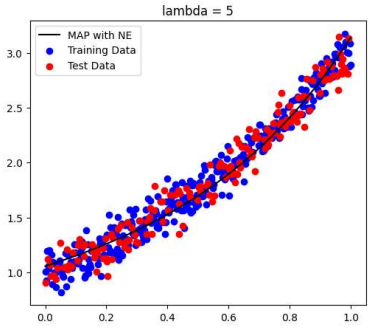

Training and test error over k
Train set error
Test set error
Error
K
2.0
1.5
1.0
0.5
0.0
2
3
4
5
6
7
8
9
10

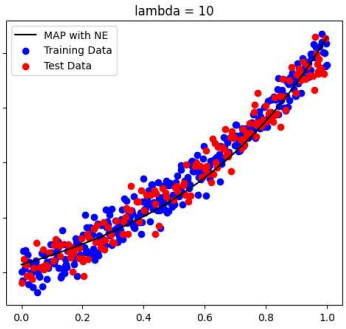
Method: NE


K = 2
lambda = 1
lambda = 5
lambda = 10
K = 3
lambda = 1
lambda = 5
lambda = 10
MAP with NE
Training Data
Test Data

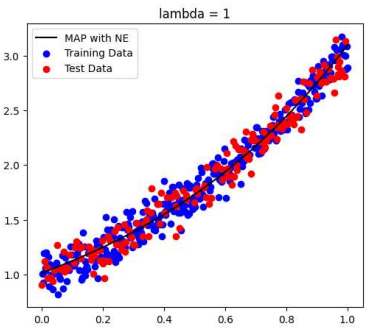
K = 6

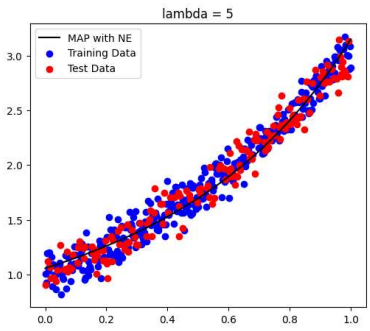


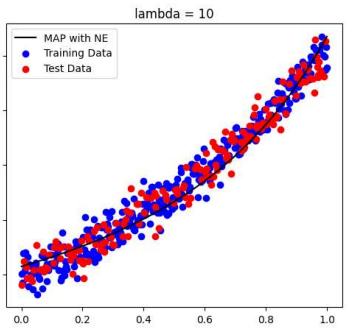




K = 8

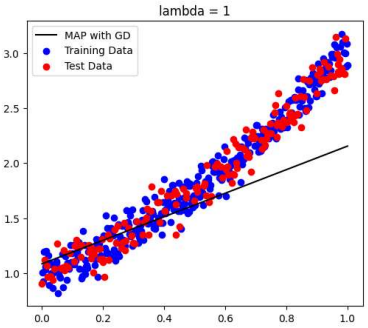


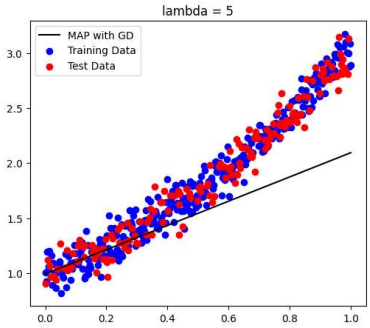


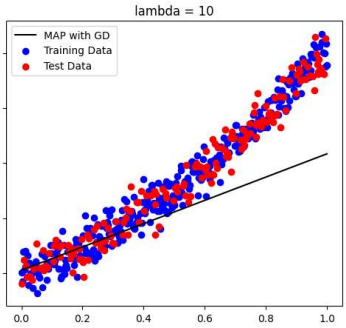


Method: GD

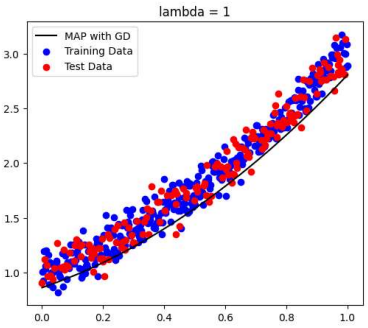
K = 2

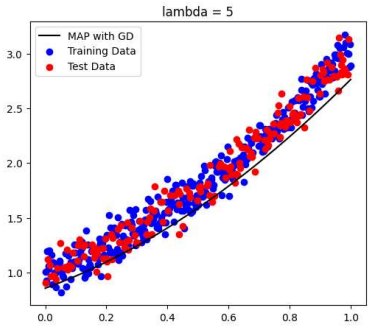


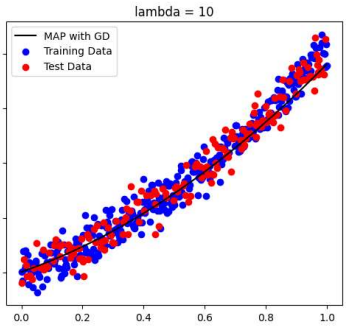




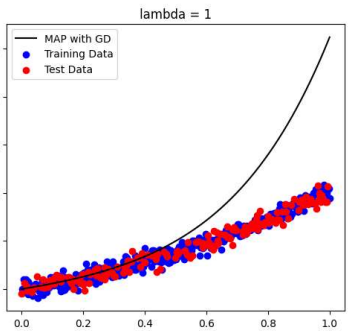
K = 3

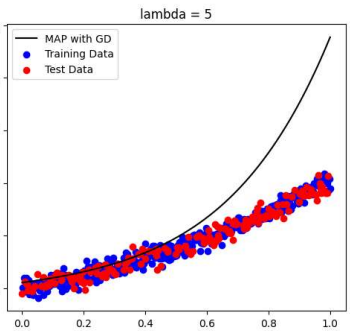


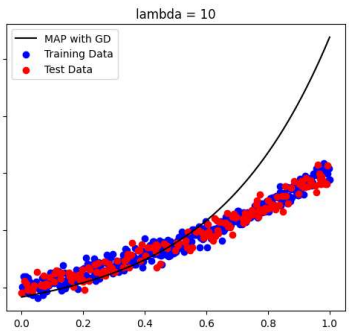




K = 6

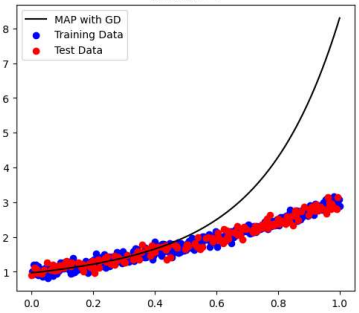




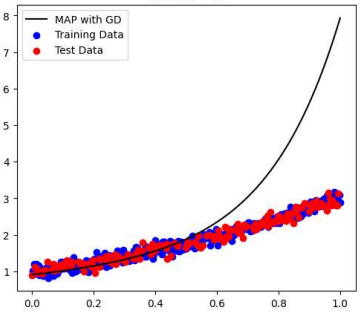


K = 8

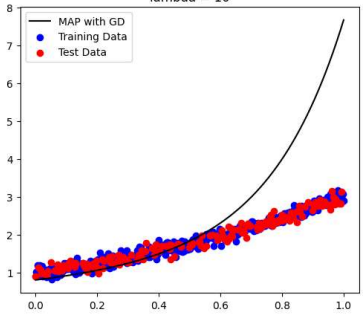
lambda = 1



lambda = 5



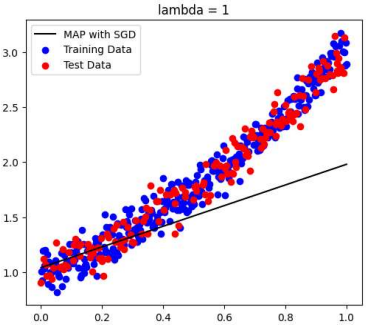
lambda = 10



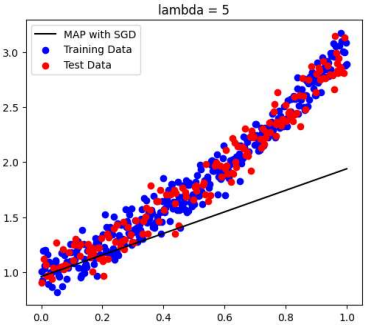
Method: SGD

K = 2

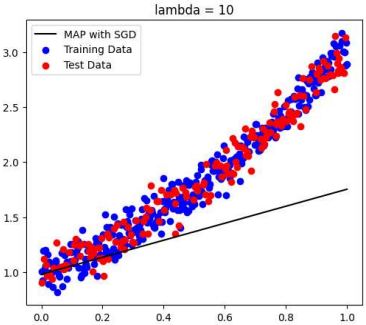
lambda = 1



lambda = 5

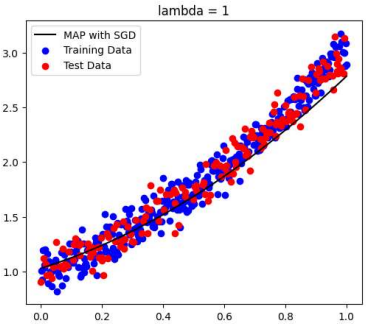


lambda = 10

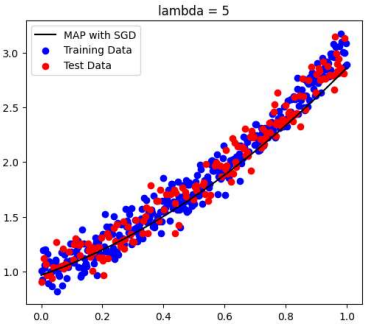


K = 3

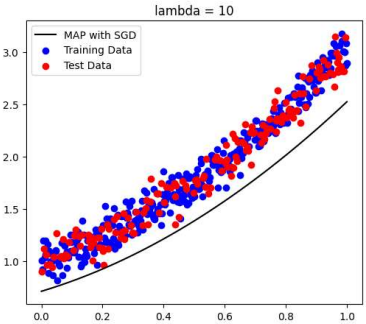
lambda = 1



lambda = 5

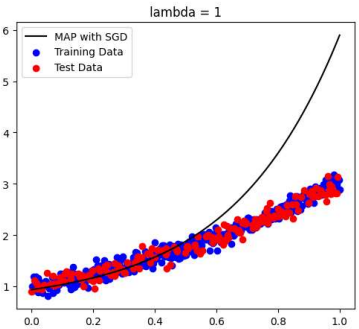


lambda = 10

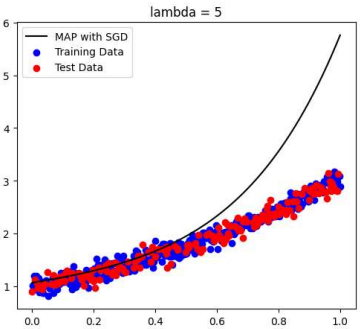


K = 6

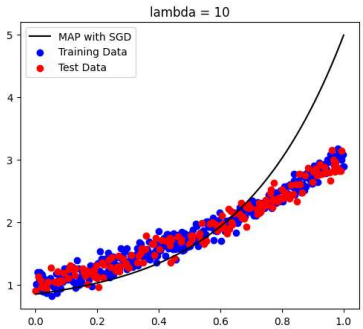
lambda = 1



lambda = 5

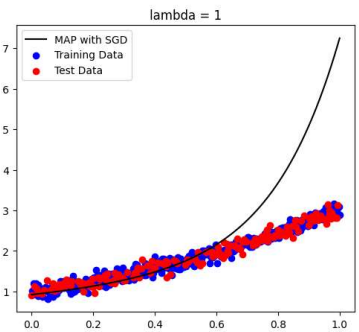


lambda = 10

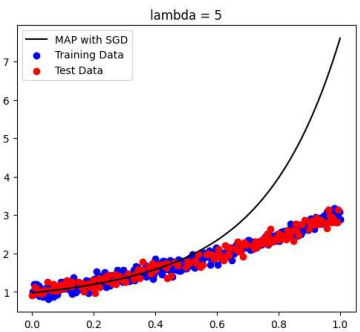


K = 8

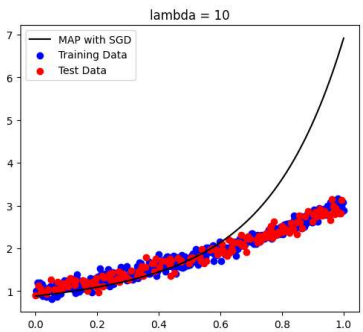
lambda = 1



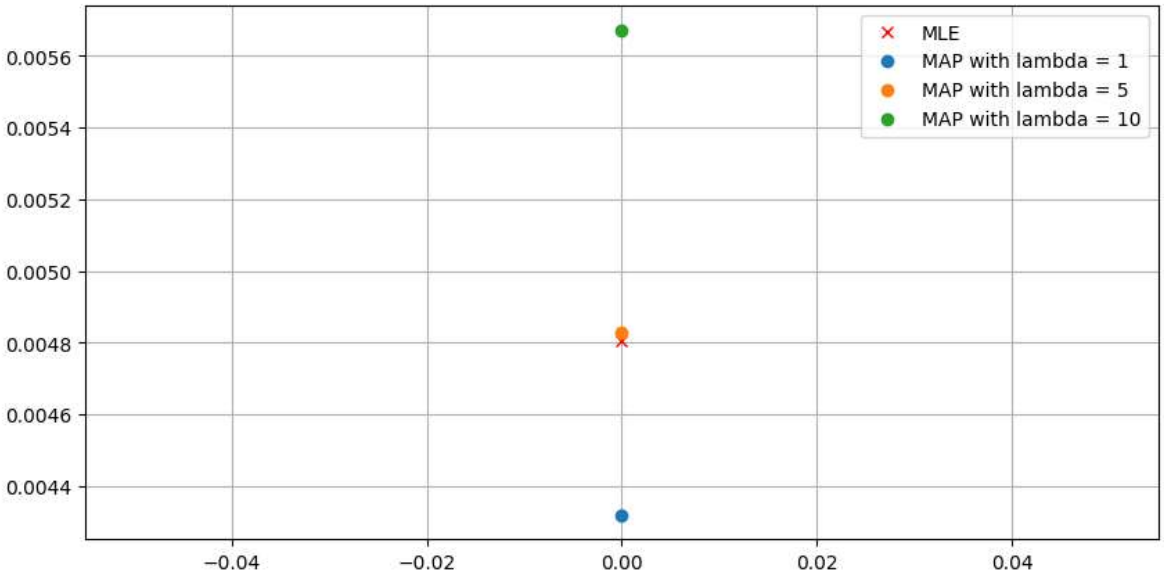
lambda = 5

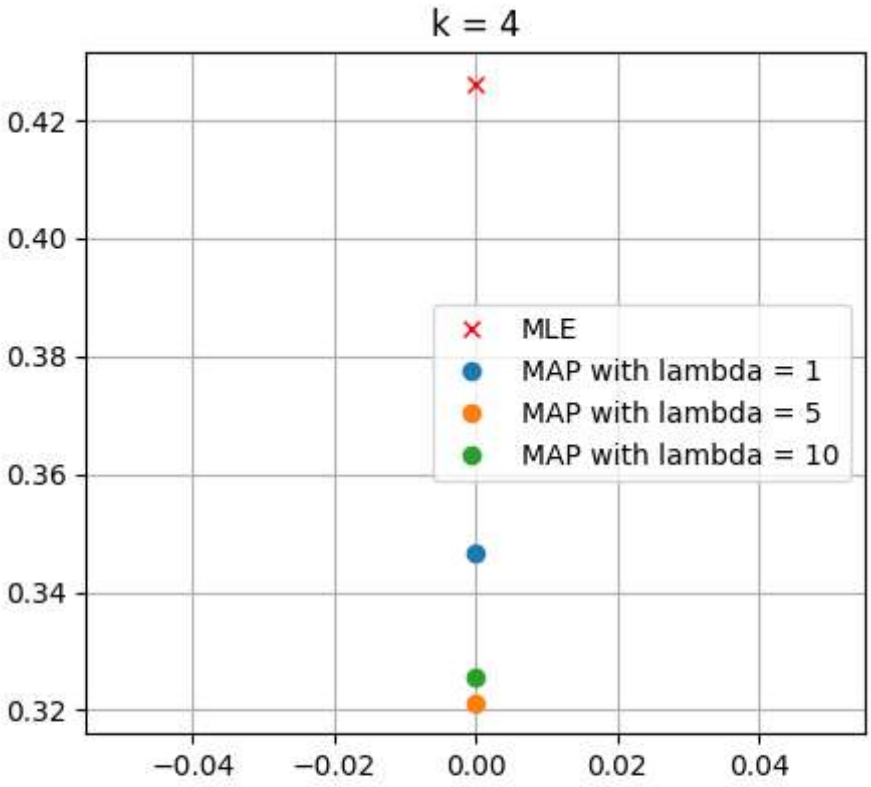


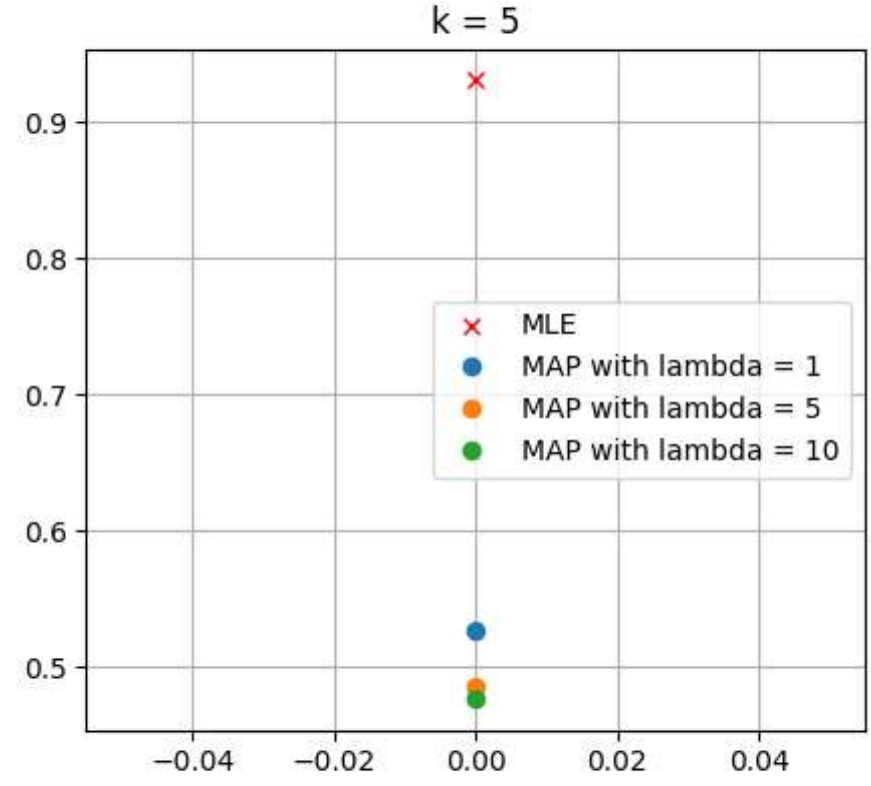
lambda = 10



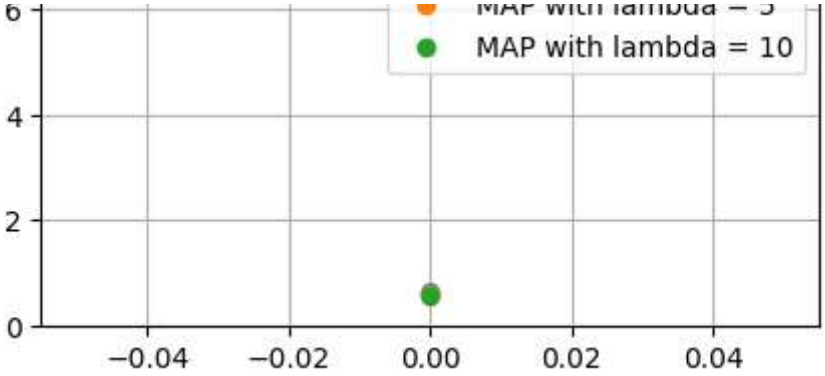
```
Theta MLE GD: [0.83070759 1.07619658 0.9197456 ]
Error MLE GD: 0.1167712979169827
Theta MLE SGD: [0.78406443 1.19156862 0.97162319]
Error MLE SGD: 0.1167712979169827
Theta MLE NE: [1.00788131 0.91361642 1.13176814]
Error MLE NE: 0.09108069687277402

Theta MAP GD: [0.93174851 0.98909649 0.99572815]
Error MAP GD: 0.03998083198123444
Theta MAP SGD: [1.05010717 0.97828494 1.05651585]
Error MAP SGD: 0.03998083198123444
Theta MAP NE: [0.99667543 1.0003787  1.02461837]
Error MAP NE: 0.014344105288132418


Method: NE


MLE test error:  0.0048037728545887925
MAP test error:  [0.00431861089543536, 0.004829826289605899, 0.00567220562614689
4]
```




k = 4
0.42
0.40
0.38
0.36
0.34
0.32
−0.04
−0.02
0.00
0.02
0.04
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10

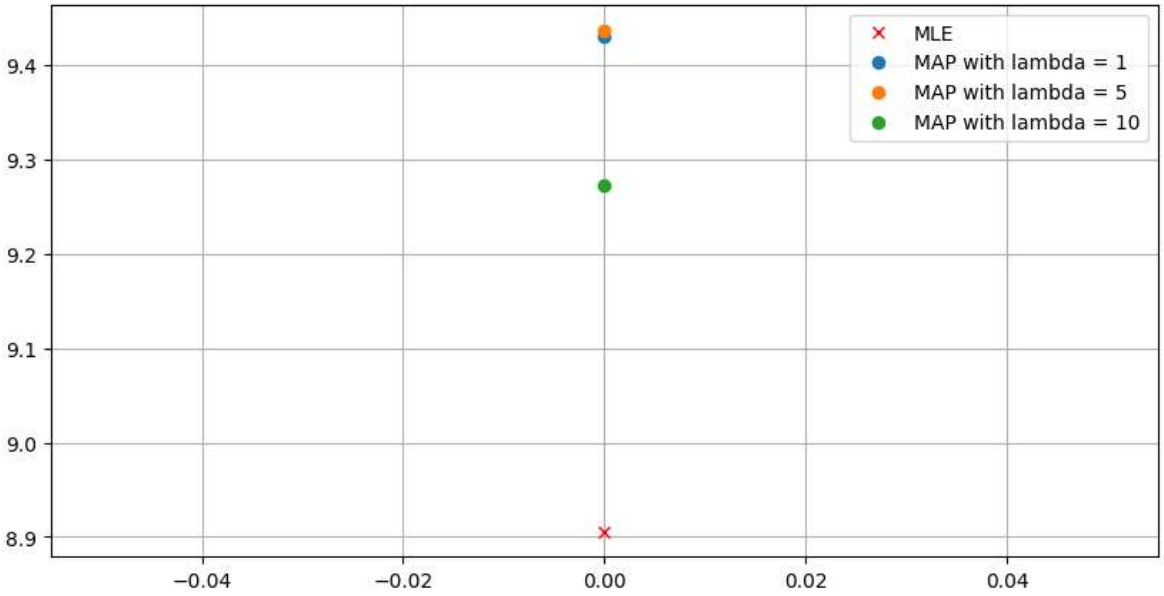

k = 5
0.9
0.8
0.7
0.6
0.5
−0.04
−0.02
0.00
0.02
0.04
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10

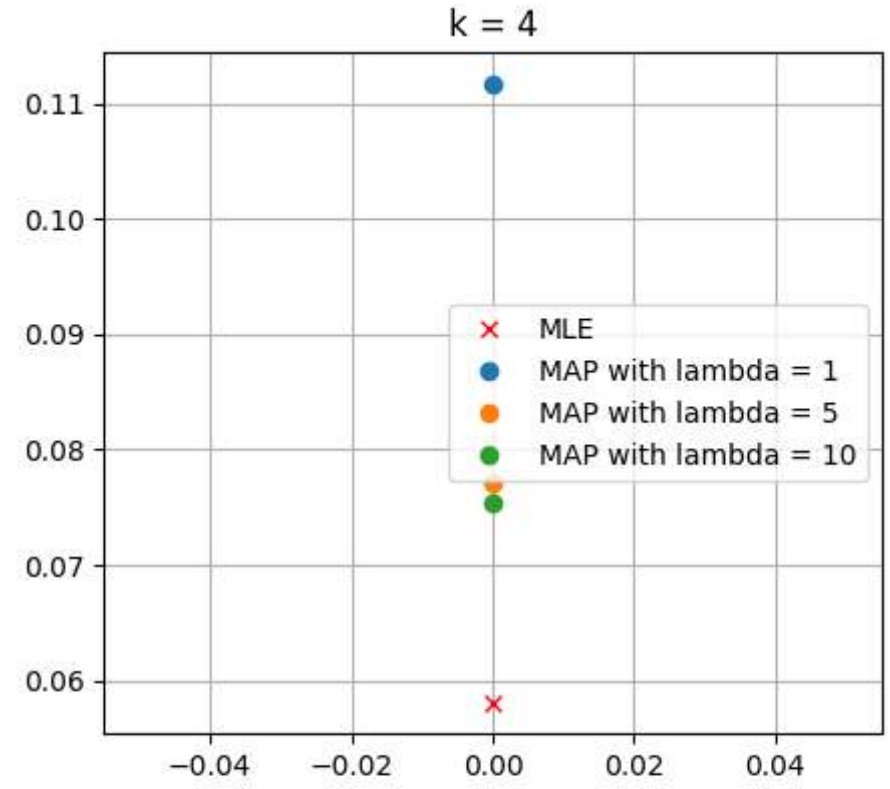

k = 6
12
10
8
MLE
MAP with lambda = 1

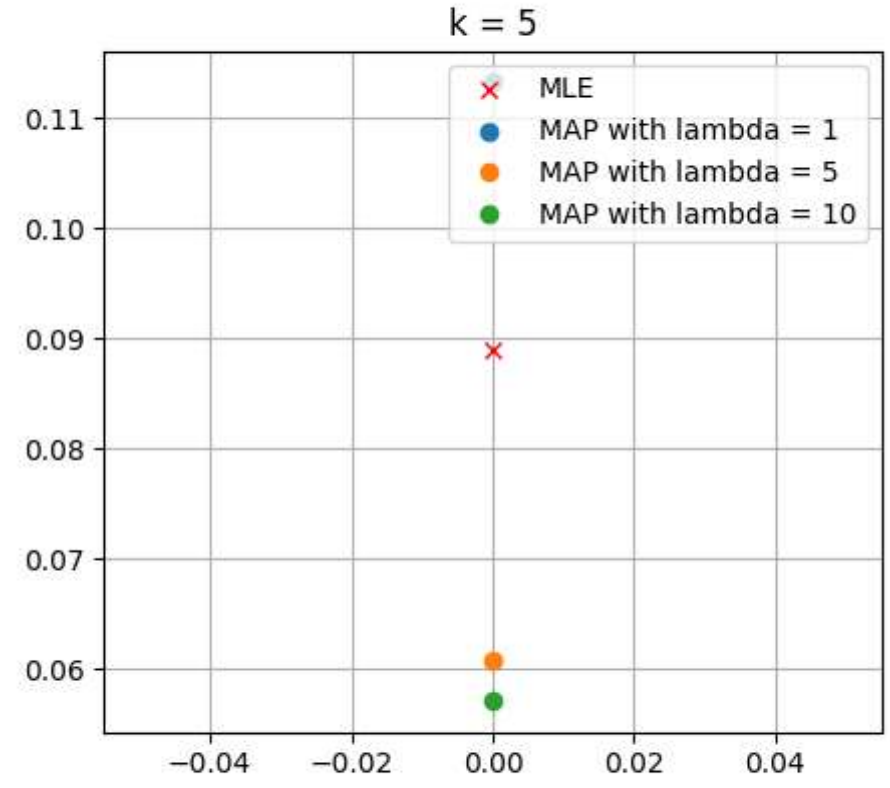


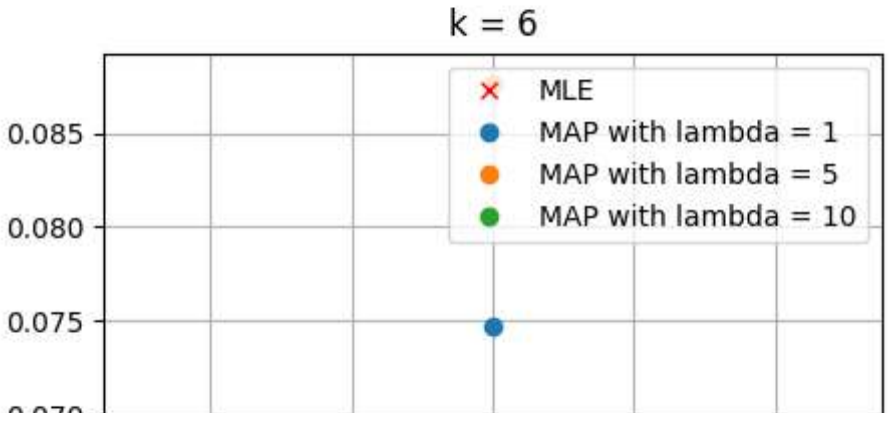
Method: GD

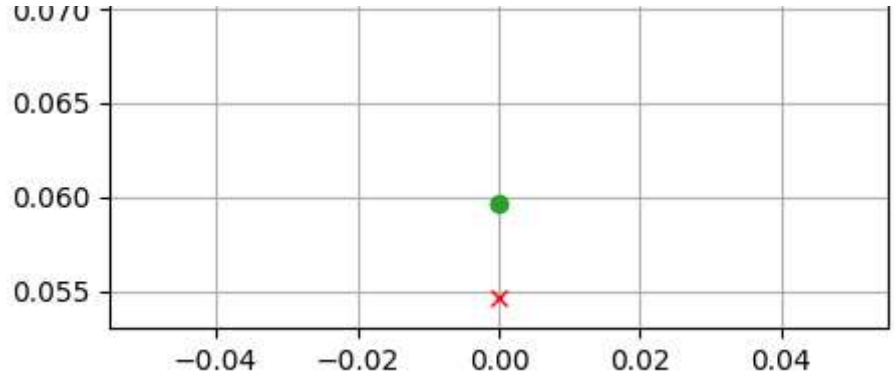
MLE test error: 8.905416403205699
MAP test error: [9.430513076442338, 9.43722001064176, 9.272199304676722]




k = 4
0.11
0.10
0.09
0.08
0.07
0.06
−0.04
−0.02
0.00
0.02
0.04
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10

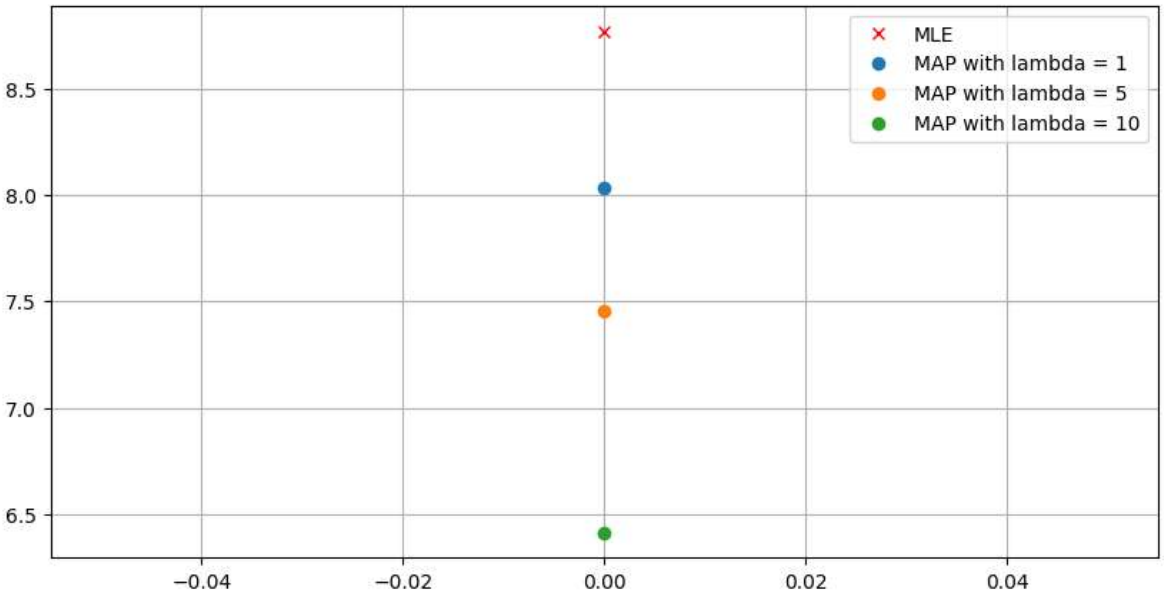

k = 5
0.11
0.10
0.09
0.08
0.07
0.06
−0.04
−0.02
0.00
0.02
0.04
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10

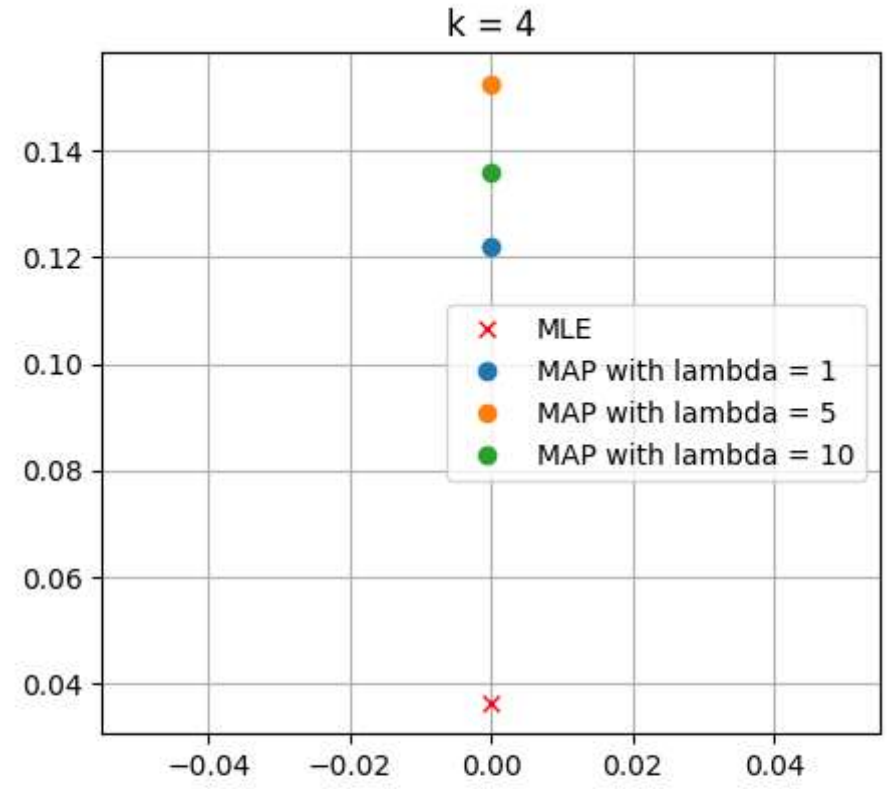

k = 6
0.085
0.080
0.075
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10

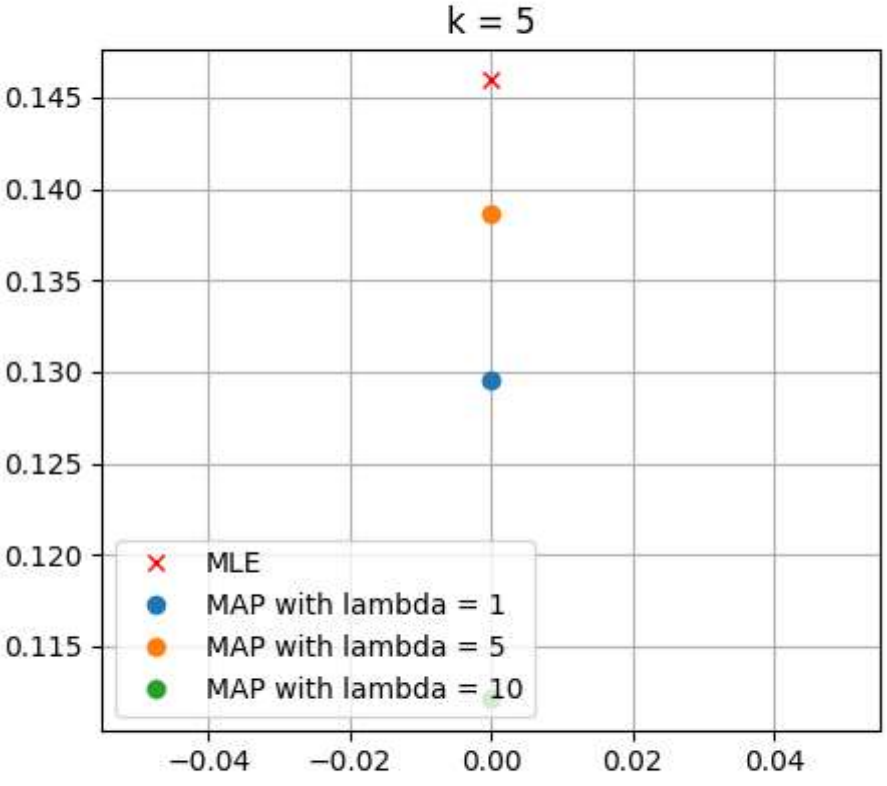


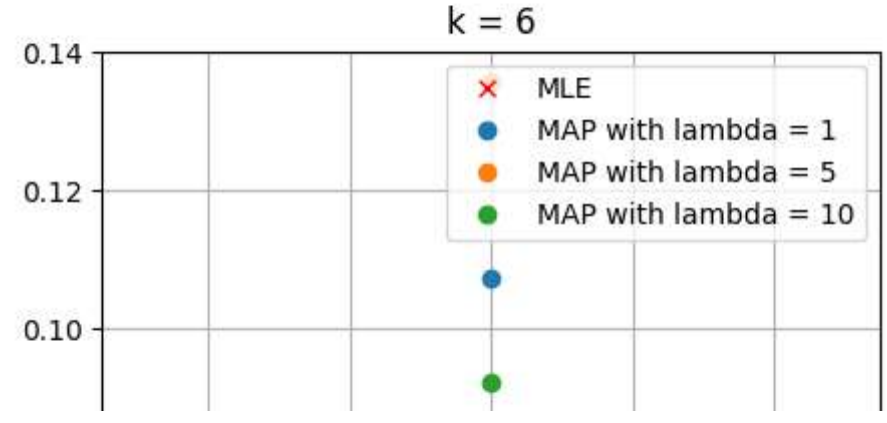
Method: SGD

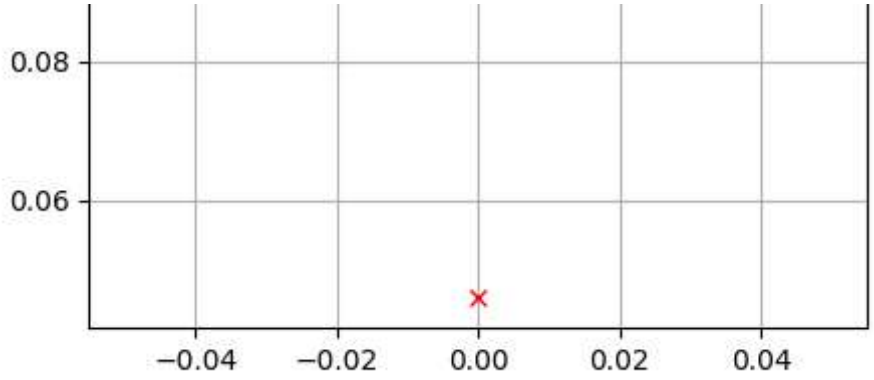
MLE test error: 8.768600603219053
MAP test error: [8.031781390517901, 7.455700036149708, 6.412769516221061]




k = 4
0.14
0.12
0.10
0.08
0.06
0.04
−0.04
−0.02
0.00
0.02
0.04
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10


k = 5
0.145
0.140
0.135
0.130
0.125
0.120
0.115
−0.04
−0.02
0.00
0.02
0.04
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10


k = 6
0.14
0.12
0.10
MLE
MAP with lambda = 1
MAP with lambda = 5
MAP with lambda = 10


0.08
0.06
−0.04
−0.02
0.00
0.02
0.04