In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt
import random
from random import randint
```

In [ ]:

```
def backtracking(f, grad_f, x):
    alpha = 1
    c = 0.8
    tau = 0.25


    while f(x - alpha * grad_f(x)) > f(x) - c * alpha * np.linalg.norm(gr
        alpha = tau * alpha

        if alpha < 1e-3:
            break
    return alpha
```

In [ ]:

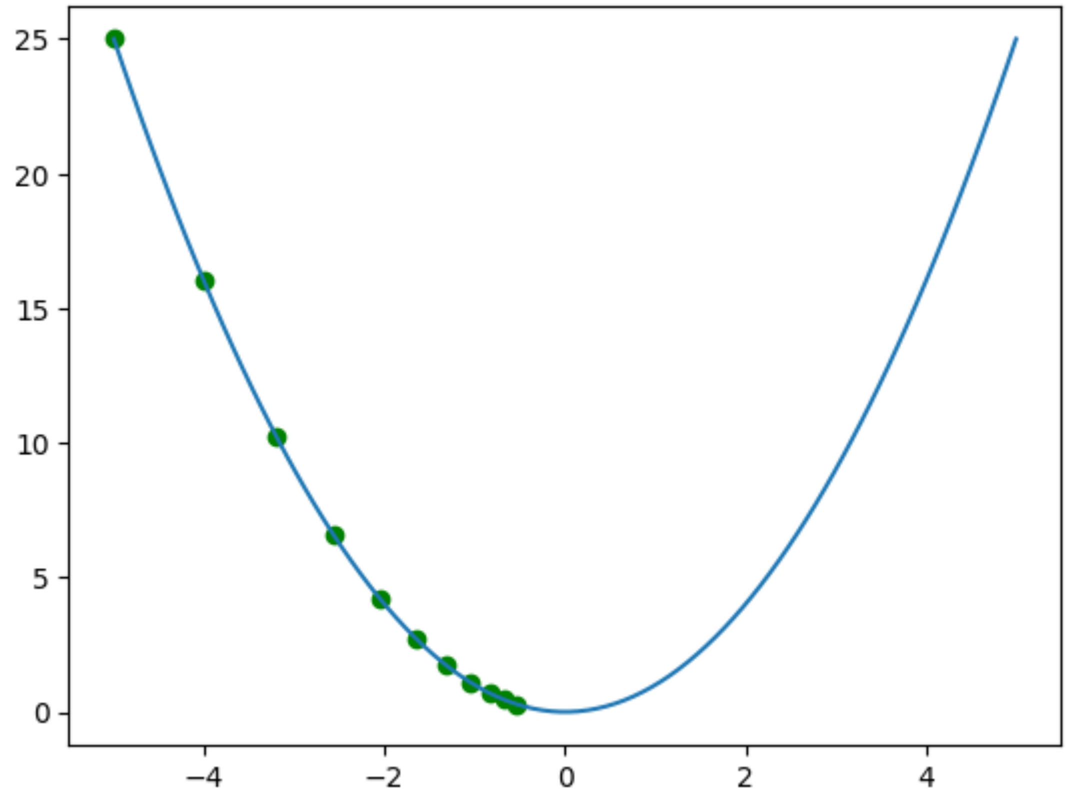
```
def gradient_descent(f, grad_f, x0, tolf, tolx, kmax, alpha= 1, bt= False
    xk = x0
    f_vals = [f(xk)]
    grad_vals = [grad_f(xk)]
    err_vals = [np.linalg.norm(grad_f(xk))]
    x_vals = [xk]
    iteration = 0

    while iteration < kmax:
        x_prec = xk

        xk = xk - alpha * grad_f(xk)

        if bt:
            alpha = backtracking(f, grad_f, xk)

        x_vals.append(xk)
        f_vals.append(f(xk))
        grad_vals.append(grad_f(xk))
        err_vals.append(np.linalg.norm(grad_f(xk)))

        iteration+=1

        if np.linalg.norm(grad_f(xk)) < tolf * np.linalg.norm(grad_f(x0))
            break

        if np.linalg.norm(xk - x_prec) < tolx * np.linalg.norm(x0):
            break

    return (x_vals, iteration, f_vals, grad_vals, err_vals)
```

In [ ]:

```
def f(x):
    return x**2

def grad_f(x):
    return 2*x

xk_vals, k, f_vals, grad_vls, err_vals = gradient_descent(f, grad_f, x0=-

x_vals = np.linspace(-5, 5, 100)
y_vals = f(x_vals)
plt.plot(x_vals, y_vals)
plt.scatter(xk_vals, f_vals, c='green')
#plt.plot(f_vals)
```

Out[ ]:

```
<matplotlib.collections.PathCollection at 0x7f4ddd09bc70>
```

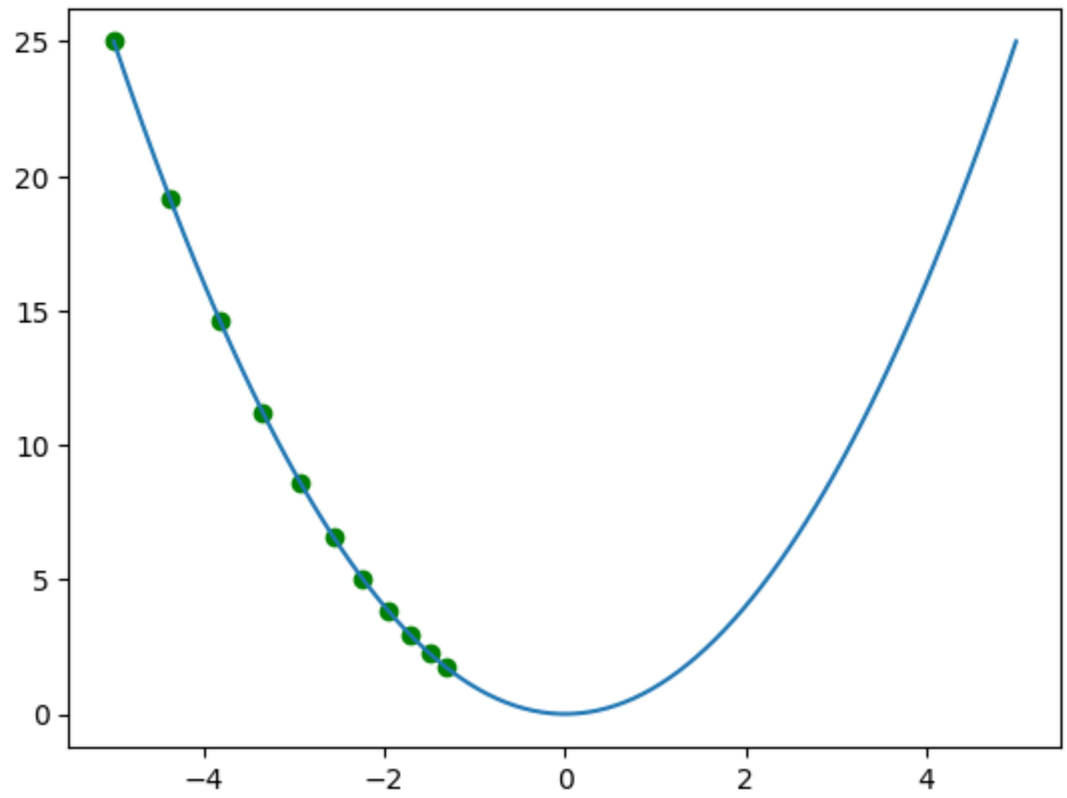


In [ ]:

```
def f(x):
    return x**2

def grad_f(x):
    return 2*x

xk_vals, k, f_vals, grad_vals, err_vals = gradient_descent(f, grad_f, x0=

x_vals = np.linspace(-5, 5, 100)
y_vals = f(x_vals)
plt.plot(x_vals, y_vals)
plt.scatter(xk_vals, f_vals, c='green')
#plt.plot(f_vals)
```

Out[ ]:

```
<matplotlib.collections.PathCollection at 0x7f4ddd0f8b50>
```



HOMEWORKING STARTING

In [ ]:

```
def f(x):
    return (2*(x[0])**2 + (x[1]-2)**2)
def grad_f(x):
    return np.array((4*x[0], 2*x[1] - 4))

def my_plot(xk_vals, k, f_vals, grad_valks, err_vals, f, title):
    xv = np.linspace(-10, 10, 100).T
    yv = np.linspace(-10, 10, 100).T

    xx,yy = np.meshgrid(xv, yv)

    zz = f([xx, yy])

    # xk_vals, k, f_vals, grad_vals, err_vals = gradient_descent(f, grad_

    xk_vals = np.array(xk_vals)

    plt.plot(xk_vals[:,0], xk_vals[:,1], '--ro')

    # plt.figure(figsize=(10, 10))
    plt.contour(xx, yy, zz)
    plt.title(title)
    plt.xlabel("x1")
    plt.ylabel("x2")
    plt.grid()
    plt.show()

def my_plot_2D(xk_vals, k, f_vals, grad_valks, err_vals, f, title):
    x_vals = np.linspace(-3, 3, 100)
    y_vals = []
    for x in x_vals:
        y_vals.append(f([x]))
    plt.plot(x_vals, y_vals)
    plt.scatter(xk_vals, f_vals, c='green')
    plt.title(title)
    plt.show()

def plot_error(iters, errs, labels, title = "Error (2-norms of gradient)"
    colors = []

    for i in range (len(iters)):
        colors.append('#%06X' % randint(0, 0xFFFFFF))

    colors = plt.get_cmap("tab20c")
    i= 0

    for item in zip(iters, errs, labels):
        plt.plot(item[0], item[1], c=colors(i/(len(iters)-1)), label = it
        i+=1
    plt.title(title)

    plt.legend(loc="upper left")
    plt.show()
```

In [ ]:

```
def function_testing(f, grad_f, x0=0, title="", alpha = 1e-1, bt = False,
    tolf = 1e-8
    tolx = 1e-8
    kmax = 100
    if not bt:
        xk_vals, k, f_vals, grad_vals, err_vals = gradient_descent(f, gra
    else:
        alpha_bt = backtracking(f, grad_f, x0)
        xk_vals, k, f_vals, grad_vals, err_vals = gradient_descent(f, gra
    if not oneD and not isMatr:
        if not bt:
            my_plot(xk_vals, k, f_vals, grad_vals, err_vals, f, title + "
        else:
            my_plot(xk_vals, k, f_vals, grad_vals, err_vals, f, title + "
    elif not isMatr:
        if not bt:
            my_plot_2D(xk_vals, k, f_vals, grad_vals, err_vals, f, title
        else:
            my_plot_2D(xk_vals, k, f_vals, grad_vals, err_vals, f, title
    print("Minimum Found =", xk_vals[k-1], "with", k, "iterations")

    to_ret = []
    if check_err:
        if not isMatr:
            for xk in xk_vals:
                to_ret.append(np.linalg.norm((xk - xtrue)))
        else:
            xtrue = np.array(xtrue)
            for xk in xk_vals:
                xk = np.array(xk)
                to_ret.append(np.linalg.norm((xk - xtrue)))

    if check_err:
        return np.arange(k+1), err_vals, to_ret
    return np.arange(k+1), err_vals
```

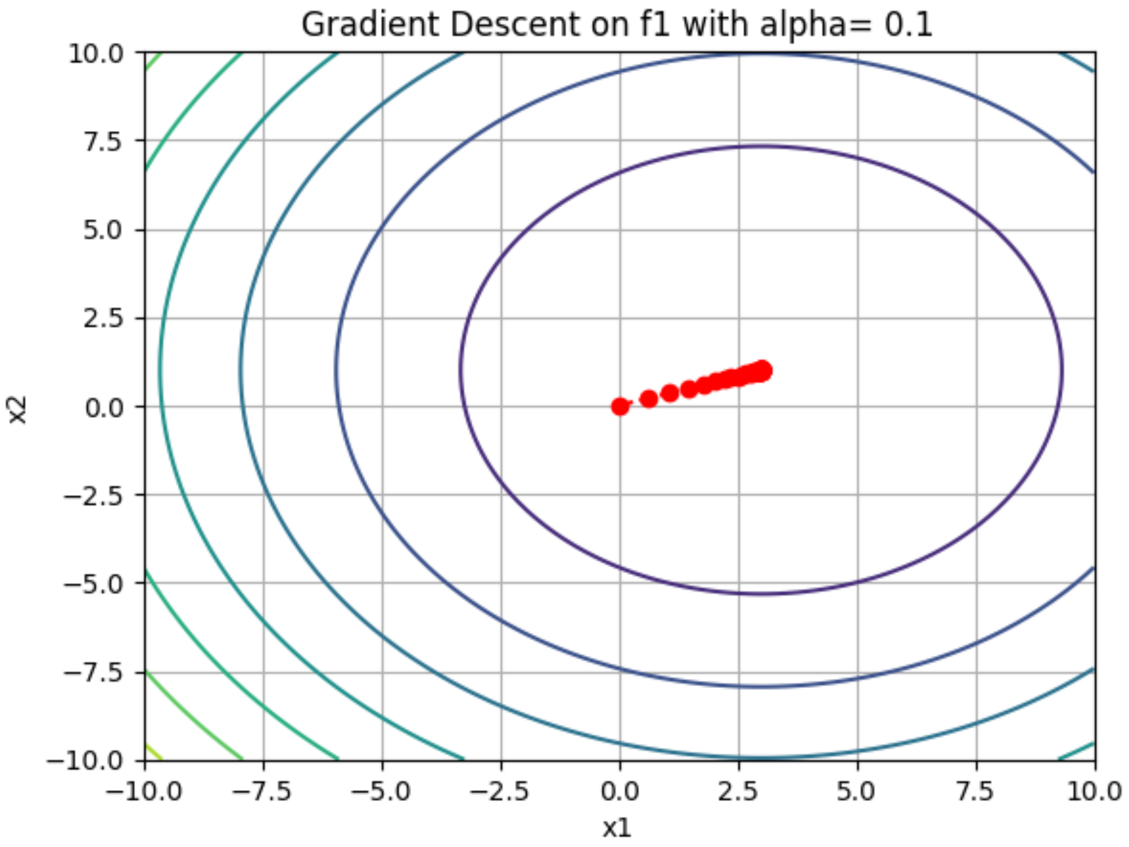
In [ ]:

```
lam = random.random()
def f1(x):
    return ((x[0] - 3)**2 + (x[1] - 1)**2)
def f2(x):
    return (10*(x[0] - 1)**2 + (x[1] - 2)**2)
def f3(x):
    x = np.array(x).T
    n = len(x)
    v = np.linspace(0, 1, n)
    A = np.vander(v)
    x_true = np.ones(n).T
    b= A @ x_true

    return ((np.linalg.norm((A @ x) - b)**2)/2)
def f4(x):
    n = len(x)
    v = np.linspace(0, 1, n)
    A = np.vander(v)
    x_true = np.ones(n).T
    b= A @ x_true

    return (((np.linalg.norm((A @ x) - b)**2)/2) + ((np.linalg.norm(x))**
def f5(x):
    return x[0]**4 + x[0]**3 - 2*(x[0]**2) - 2*x[0]
```

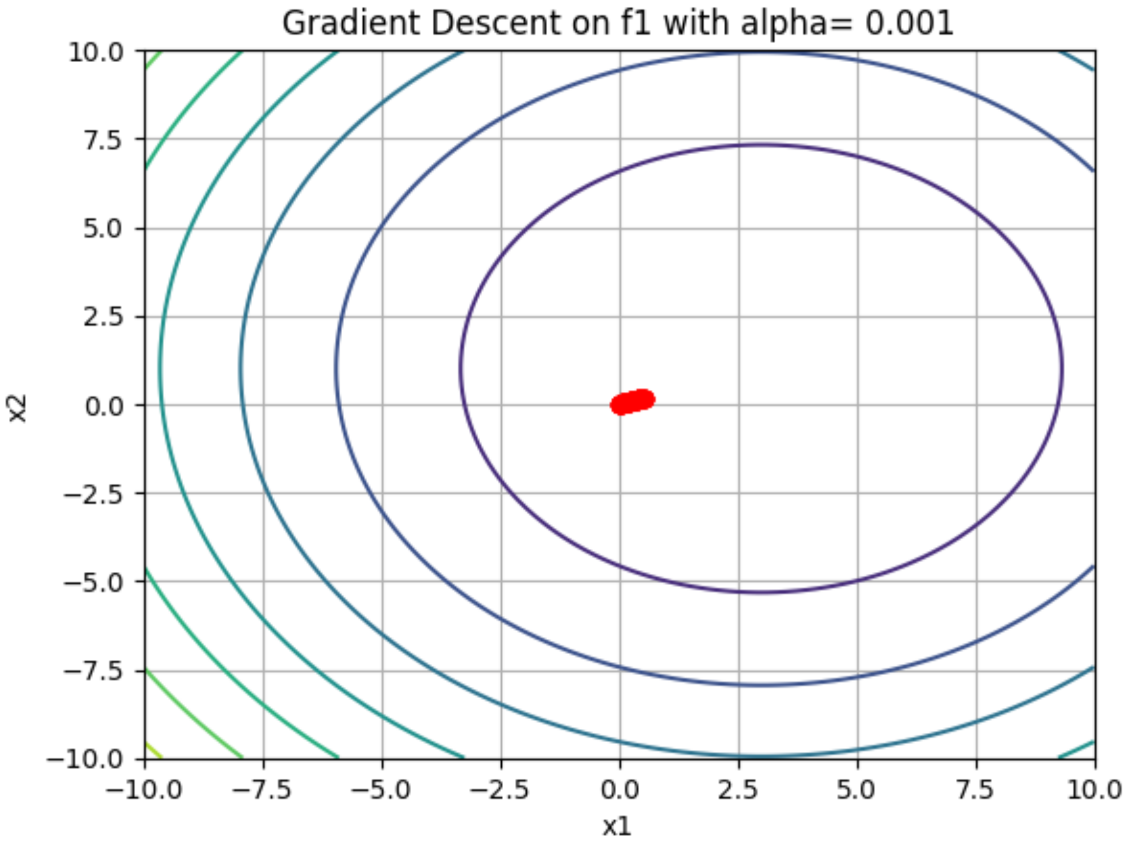
In [ ]:

```
def grad_f1(x):
    return np.array((2*x[0] - 6, 2*x[1] - 2))
def grad_f2(x):
    return np.array(20*(x[0] - 1), 2*(x[1] - 2))
def grad_f3(x):
    n = len(x)
    v = np.linspace(0,1,n)
    A = np.vander(v)
    x_true = np.ones(n).T
    b = A @ x_true
    return A.T@(A@x-b)
def grad_f4(x):
    return grad_f3(x) + lam*np.array(x)
def grad_f5(x):
    return np.array(4*x[0]**3 + 3*x[0]**2 - 4*x[0] - 2)
```

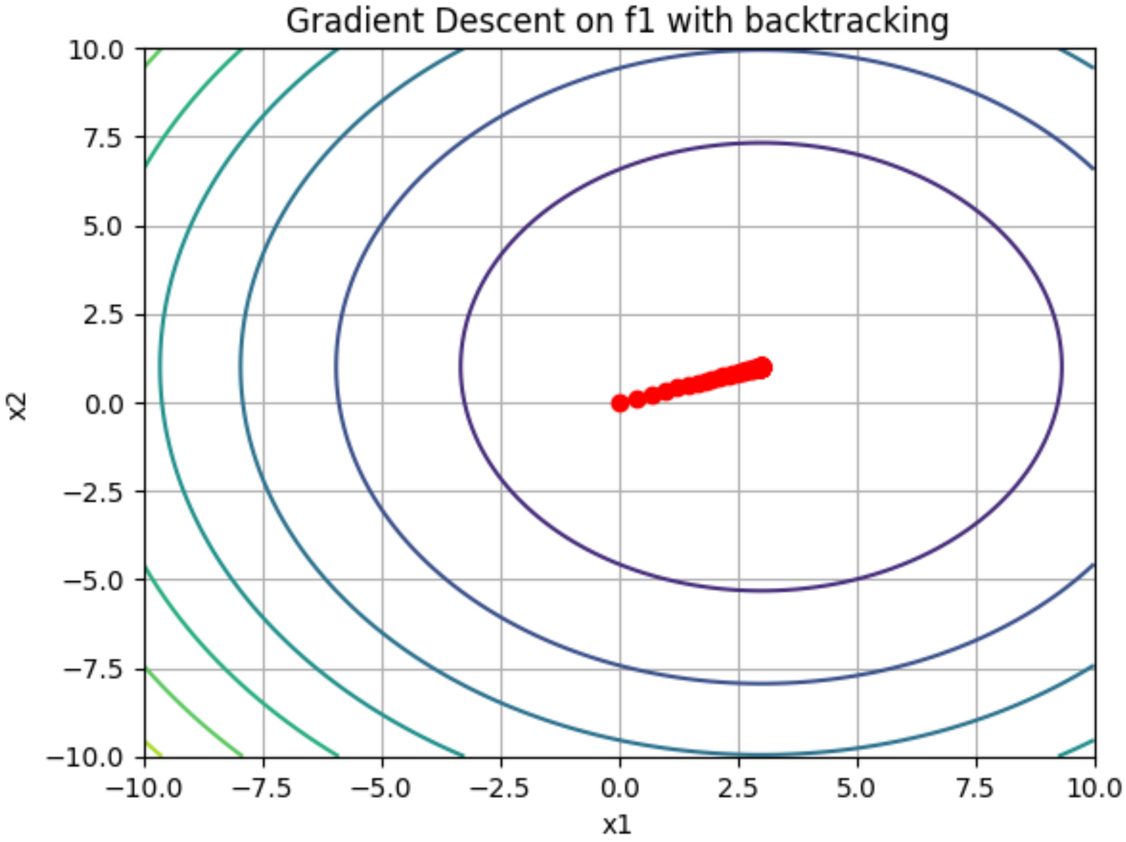
In [ ]:
```
iters = []
err_vals = []
labels = []
err_xtrue = []
xtrue = np.array([3,1]).T

el1, el2, el3 = function_testing(f1, grad_f1, x0 = np.array((0, 0)), titl
iters.append(el1)
err_vals.append(el2)
labels.append("Alpha = 1e-1")
err_xtrue.append(el3)

el1, el2, el3 = function_testing(f1, grad_f1, x0 = np.array((0, 0)), titl
iters.append(el1)
err_vals.append(el2)
labels.append("Alpha = 1e-3")
err_xtrue.append(el3)

el1, el2, el3 = function_testing(f1, grad_f1, x0 = np.array((0, 0)), titl
iters.append(el1)
err_vals.append(el2)
labels.append("Backtracking")
err_xtrue.append(el3)

plot_error(iters, err_vals, labels)
plot_error(iters, err_xtrue, labels, title="Error (distance from x_true)"
```



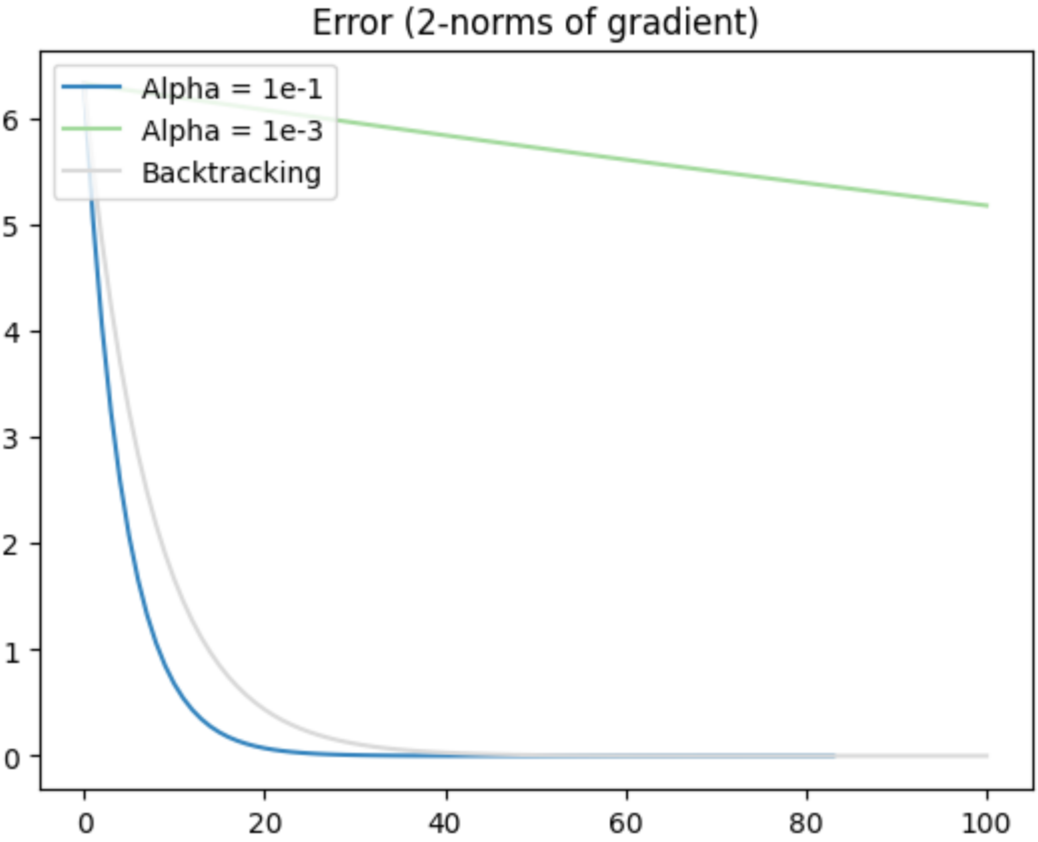
```
Minimum Found = [2.99999997 0.99999999] with 83 iterations
```

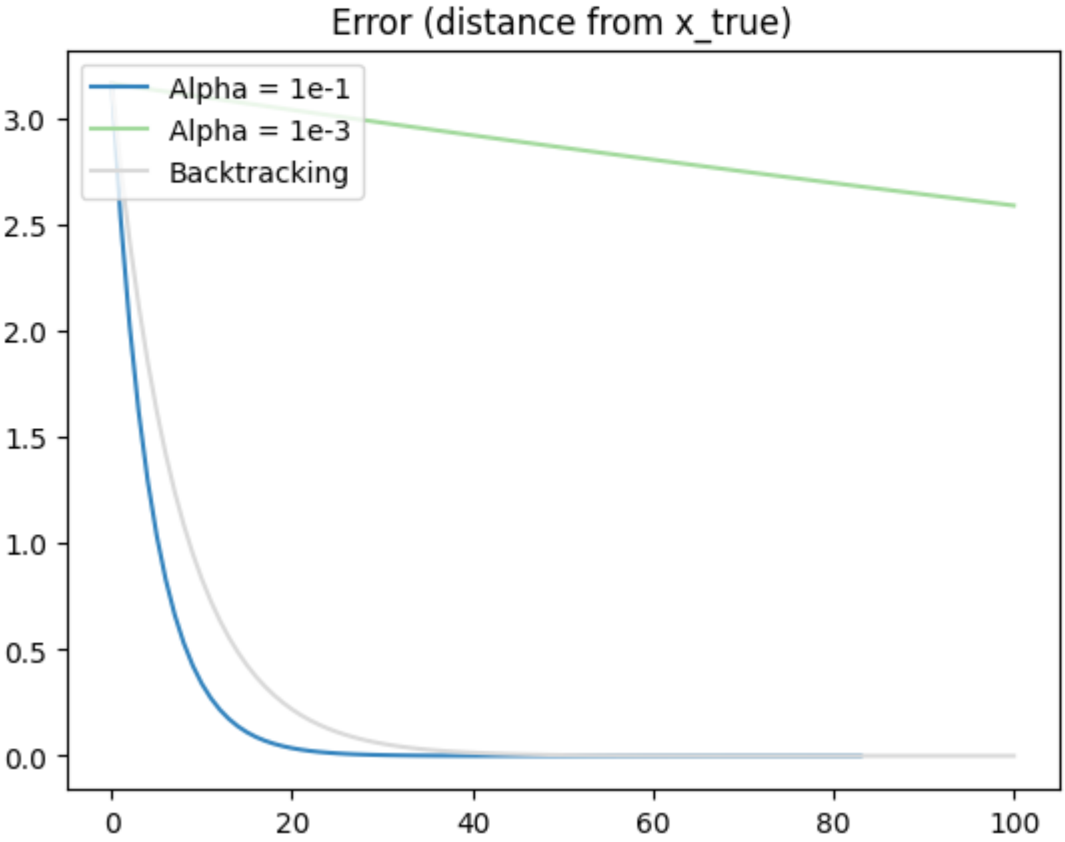


```
Minimum Found = [0.53937834 0.17979278] with 100 iterations
```



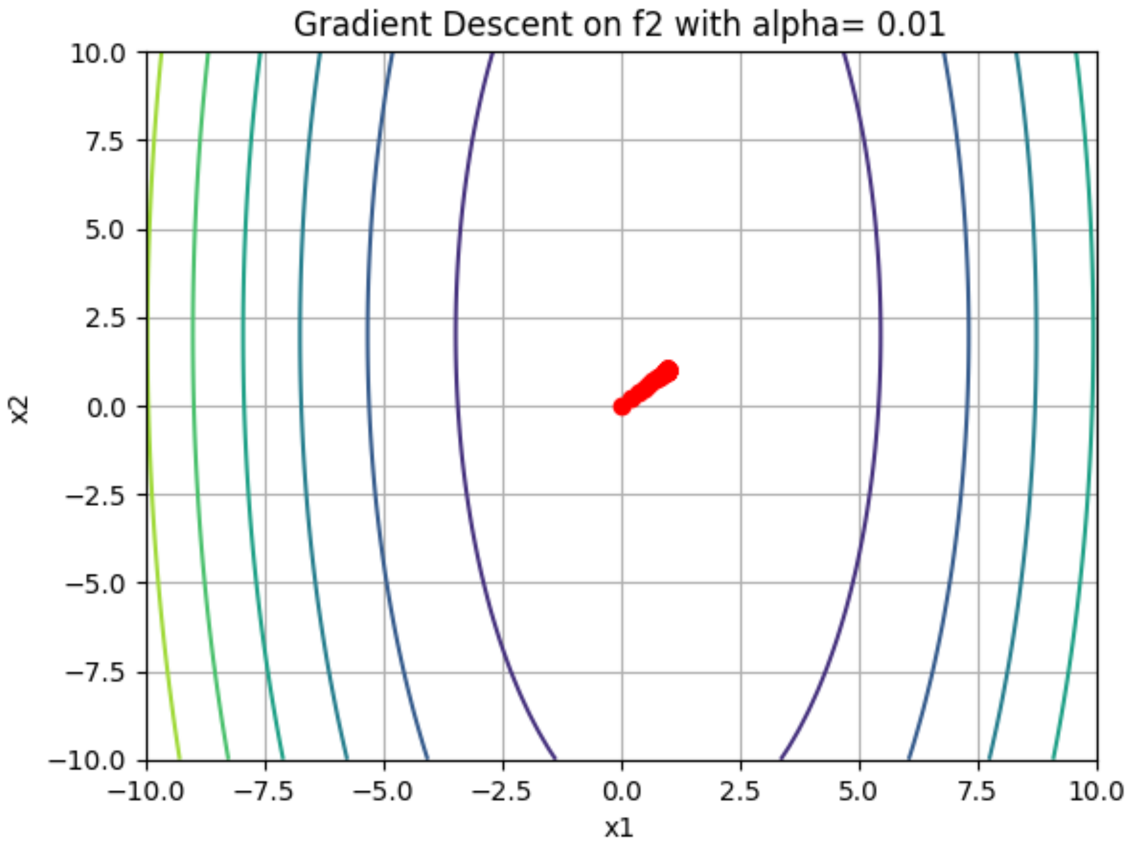
```
Minimum Found = [2.99999456 0.99999819] with 100 iterations
```


Error (2-norms of gradient)
Alpha = 1e-1
Alpha = 1e-3
Backtracking
6
5
4
3
2
1
0
0
20
40
60
80
100

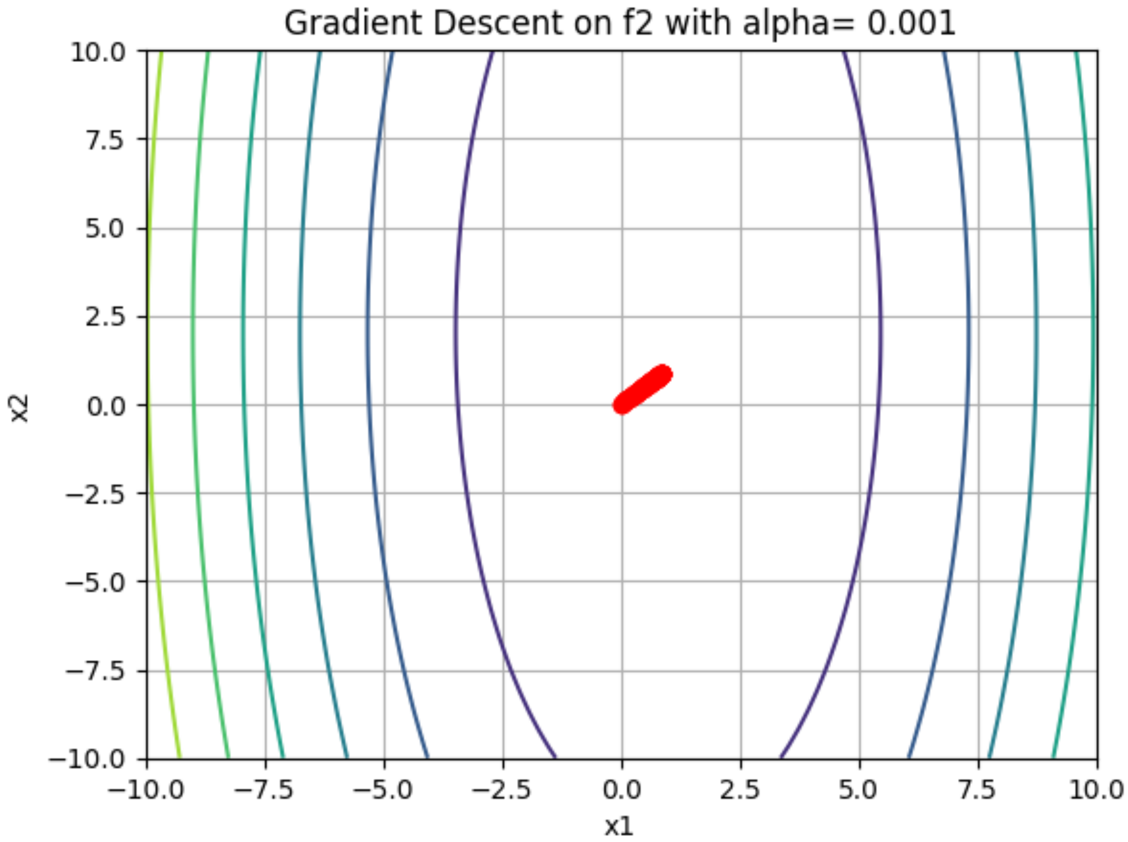

Error (distance from x_true)
Alpha = 1e-1
Alpha = 1e-3
Backtracking
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
20
40
60
80
100

In [ ]:

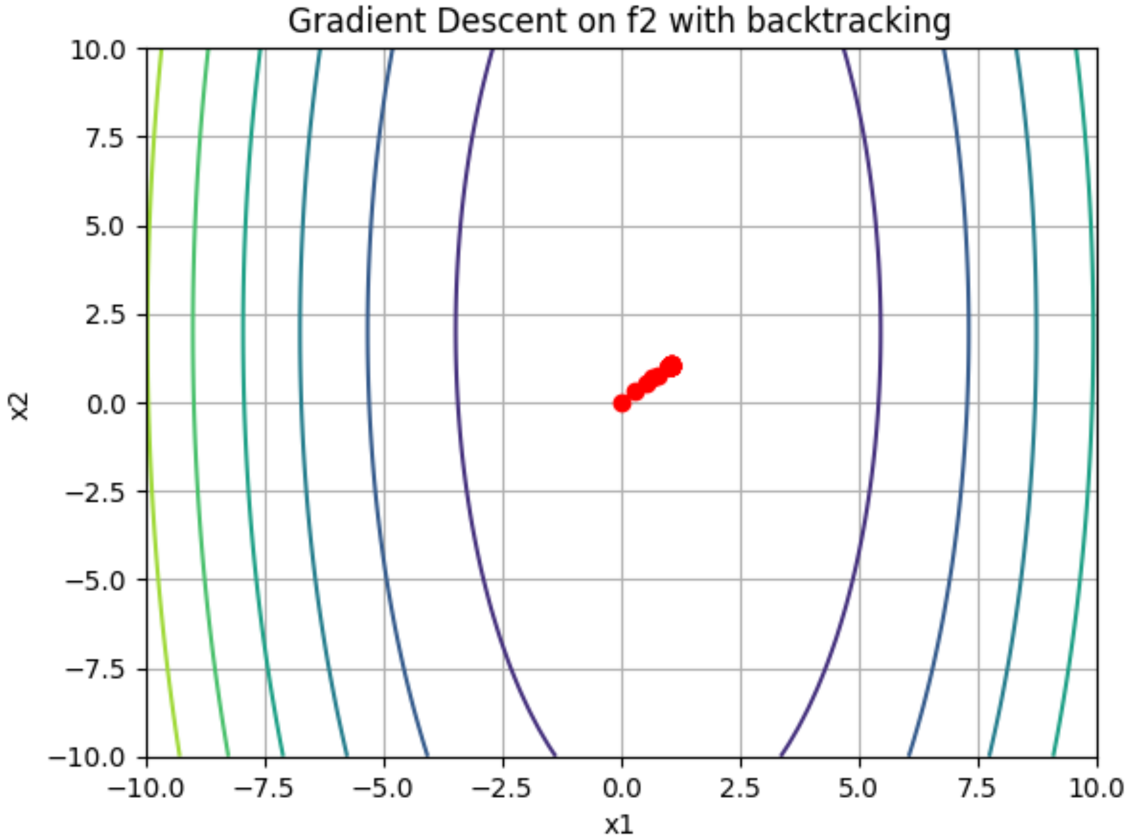
```
iters = []
err_vals = []
labels = []
err_xtrue = []
xtrue = np.array([1,2]).T

el1, el2, el3 = function_testing(f2, grad_f2, x0 = np.array((0, 0)), titl
iters.append(el1)
err_vals.append(el2)
labels.append("Alpha = 1e-1")
err_xtrue.append(el3)

el1, el2, el3 = function_testing(f2, grad_f2, x0 = np.array((0, 0)), titl
iters.append(el1)
err_vals.append(el2)
labels.append("Alpha = 1e-3")
err_xtrue.append(el3)

el1, el2, el3 = function_testing(f2, grad_f2, x0 = np.array((0, 0)), titl
iters.append(el1)
err_vals.append(el2)
labels.append("Backtracking")
err_xtrue.append(el3)

plot_error(iters, err_vals, labels)
plot_error(iters, err_xtrue, labels)
```



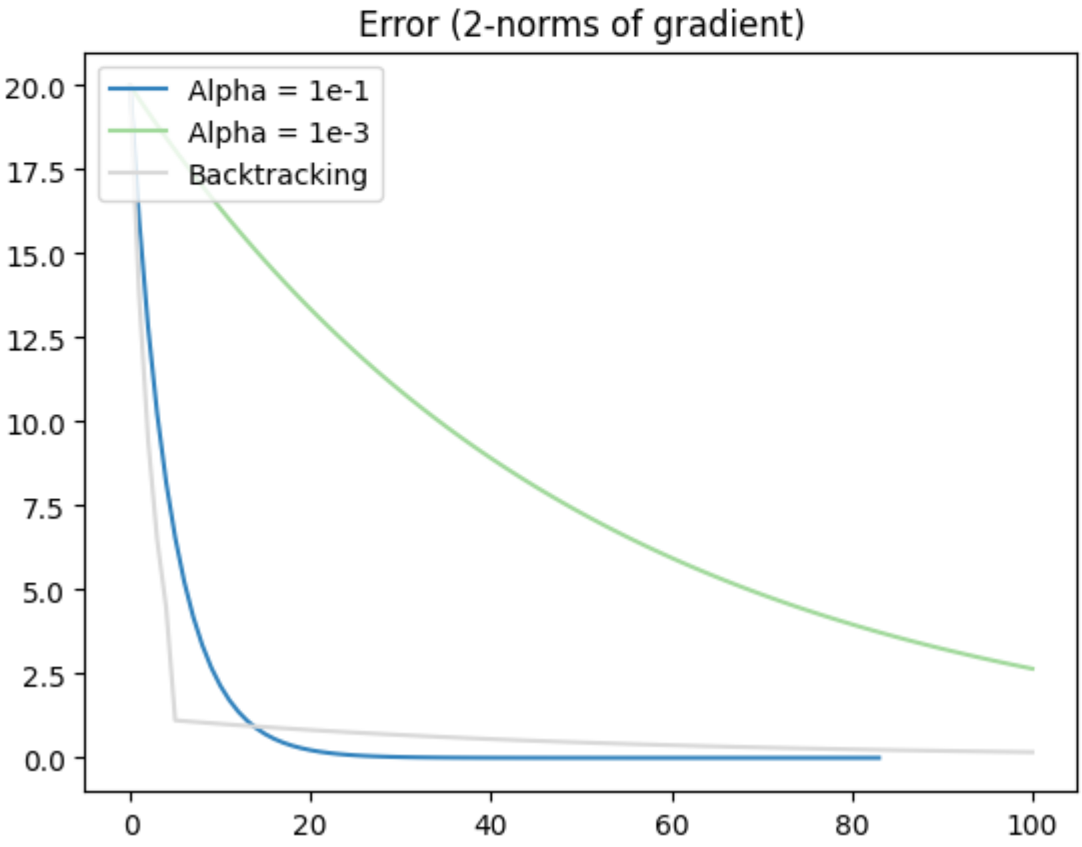
```
Minimum Found = [0.99999999 0.99999999] with 83 iterations
```

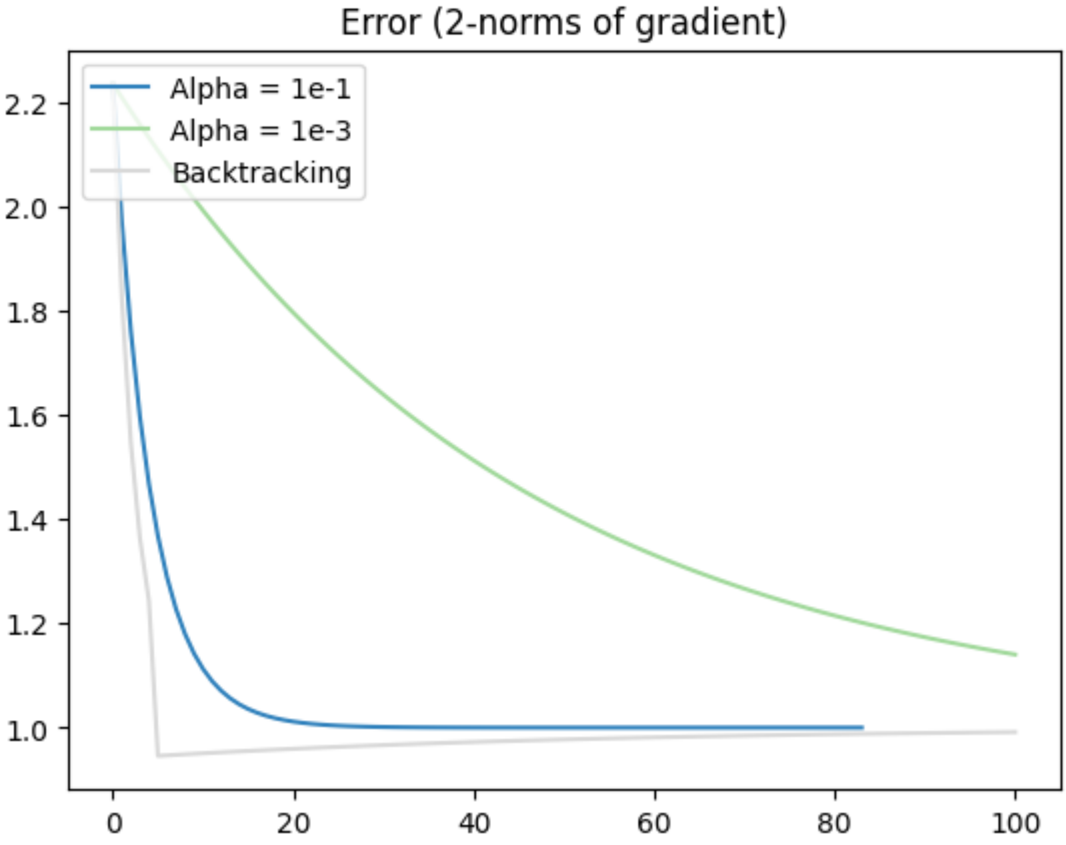


```
Minimum Found = [0.86467392 0.86467392] with 100 iterations
```

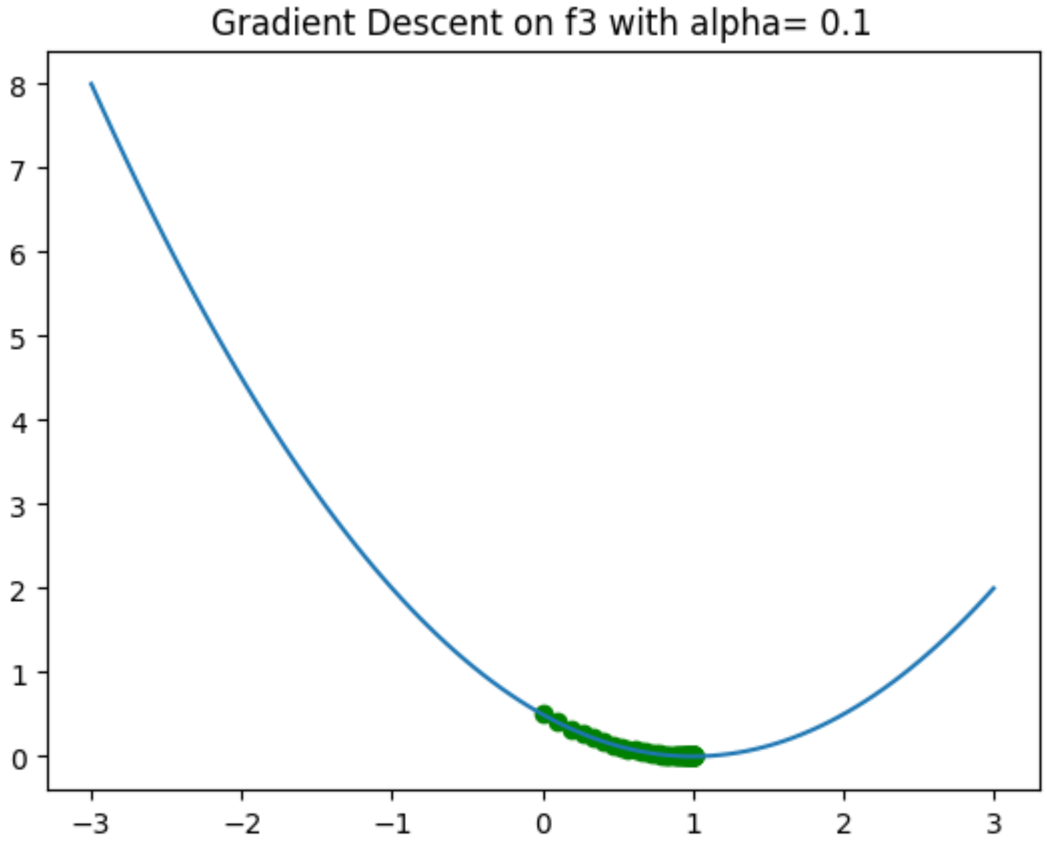


```
Minimum Found = [1.00874589 1.00874589] with 100 iterations
```

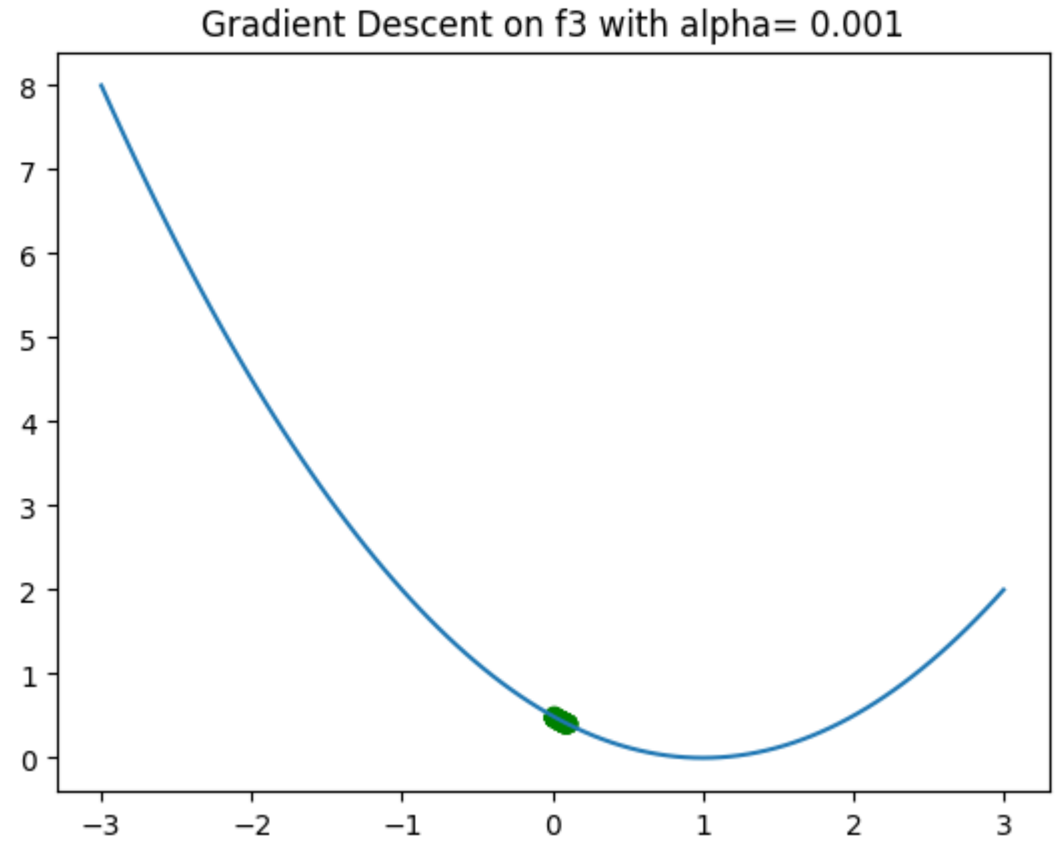

Error (2-norms of gradient)
Alpha = 1e-1
Alpha = 1e-3
Backtracking
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
20
40
60
80
100


Error (2-norms of gradient)
Alpha = 1e-1
Alpha = 1e-3
Backtracking
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0
20
40
60
80
100

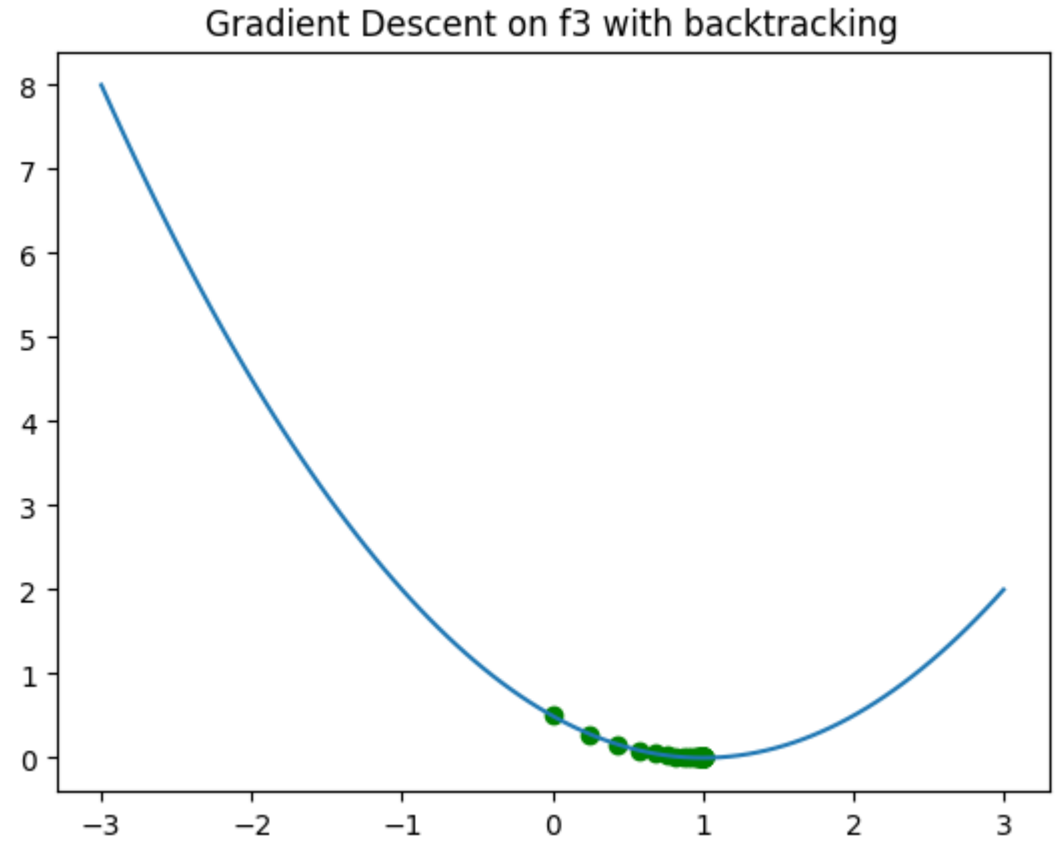
```
In [ ]: iters = []
        err_vals = []
        labels = []
        err_xtrue = []
        xtrue = np.ones(1).T

        el1, el2, el3 = function_testing(f3, grad_f3, x0 = [0], title="Gradient D
        iters.append(el1)
        err_vals.append(el2)
        labels.append("Alpha = 1e-1")
        err_xtrue.append(el3)

        el1, el2, el3 = function_testing(f3, grad_f3, x0 = [0], title="Gradient D
        iters.append(el1)
        err_vals.append(el2)
        labels.append("Alpha = 1e-3")
        err_xtrue.append(el3)

        el1, el2, el3 = function_testing(f3, grad_f3, x0 = [0], title="Gradient D
        iters.append(el1)
        err_vals.append(el2)
        labels.append("Backtracking")
        err_xtrue.append(el3)

        plot_error(iters, err_vals, labels)
        plot_error(iters, err_xtrue, labels, title="Error (distance from x_true)"
```



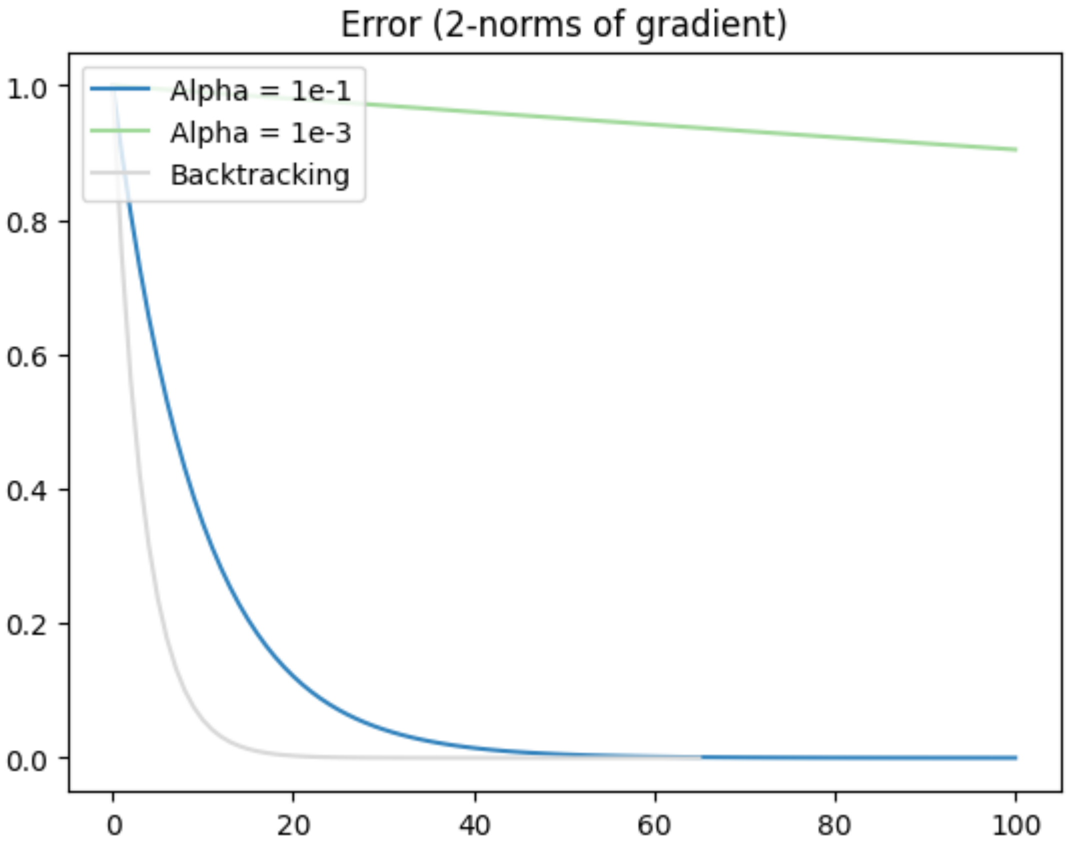
```
Minimum Found = [0.99997049] with 100 iterations
```

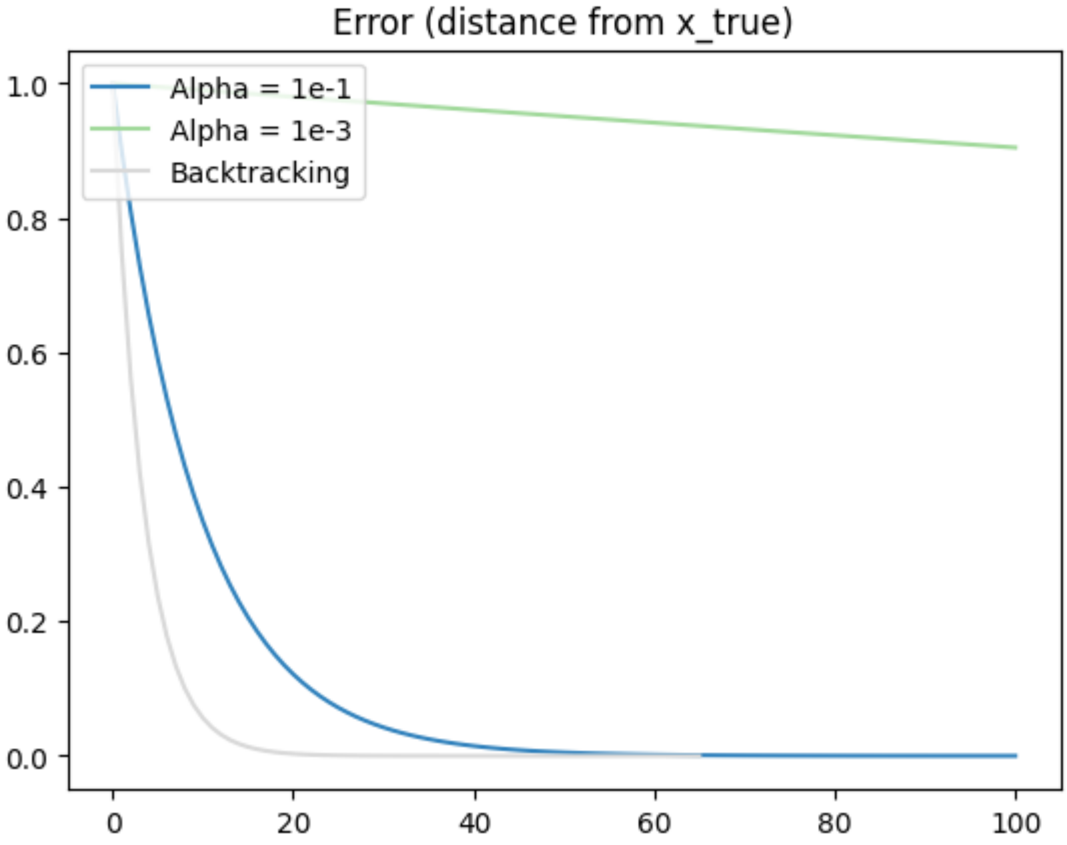


```
Minimum Found = [0.09430216] with 100 iterations
```



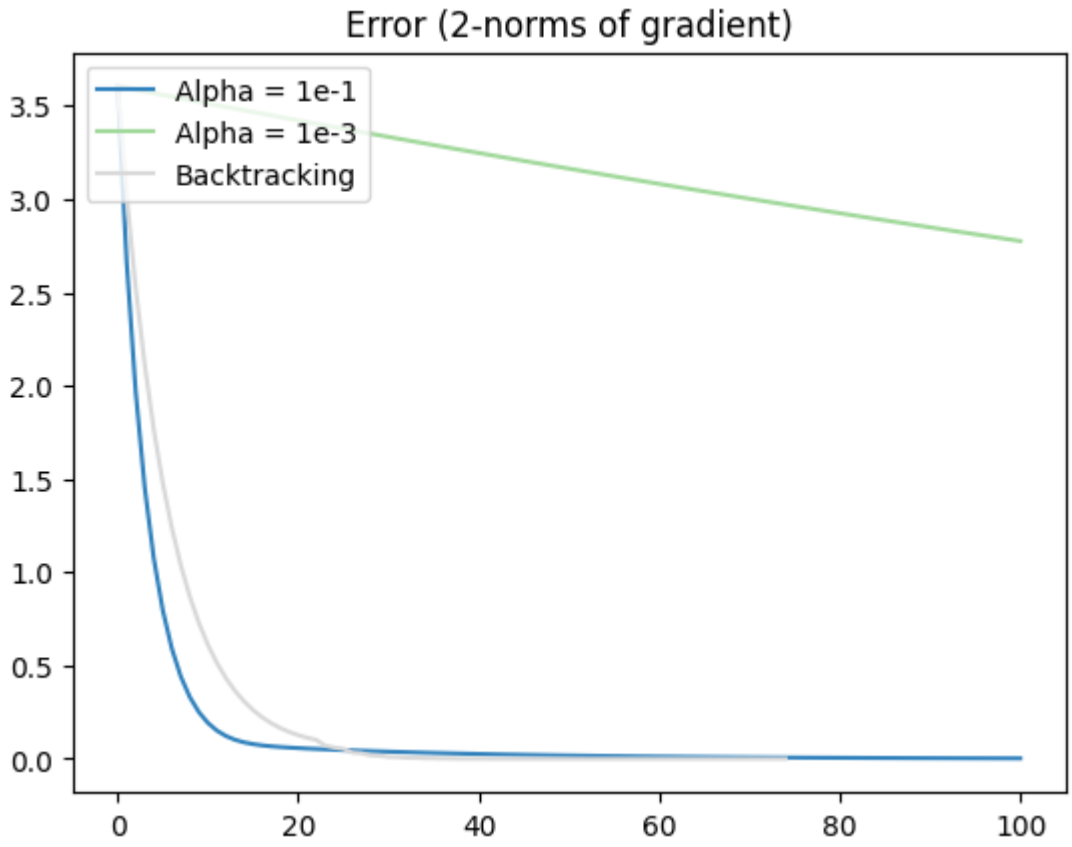
```
Minimum Found = [0.99999999] with 65 iterations
```

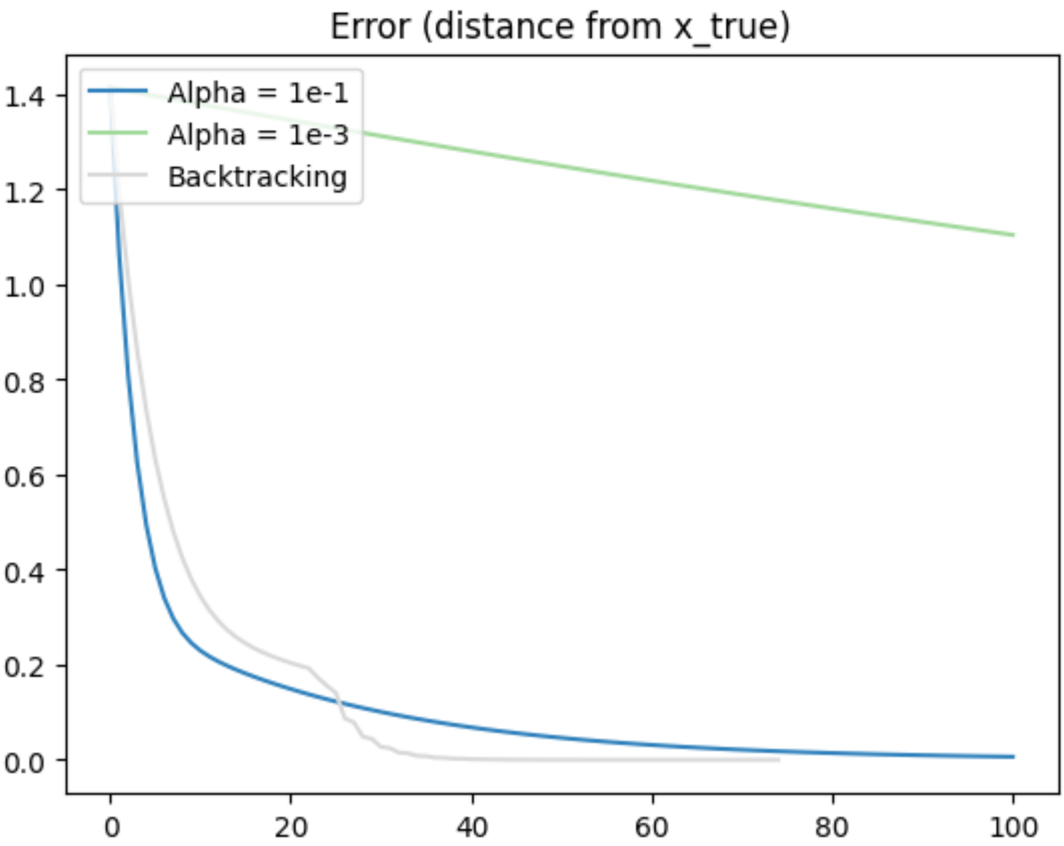

Error (2-norms of gradient)
Alpha = 1e-1
Alpha = 1e-3
Backtracking
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100


Error (distance from x_true)
Alpha = 1e-1
Alpha = 1e-3
Backtracking
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

```
In [ ]: iters = []
        err_vals = []
        labels = []
        err_xtrue = []
        xtrue = np.ones(2).T

        el1, el2, el3 = function_testing(f3, grad_f3, x0 = [0, 0], title="Gradien
        iters.append(el1)
        err_vals.append(el2)
        labels.append("Alpha = 1e-1")
        err_xtrue.append(el3)

        el1, el2, el3 = function_testing(f3, grad_f3, x0 = [0, 0], title="Gradien
        iters.append(el1)
        err_vals.append(el2)
        labels.append("Alpha = 1e-3")
        err_xtrue.append(el3)

        el1, el2, el3 = function_testing(f3, grad_f3, x0 = [0, 0], title="Gradien
        iters.append(el1)
        err_vals.append(el2)
        labels.append("Backtracking")
        err_xtrue.append(el3)

        plot_error(iters, err_vals, labels)
        plot_error(iters, err_xtrue, labels, title="Error (distance from x_true)"
```

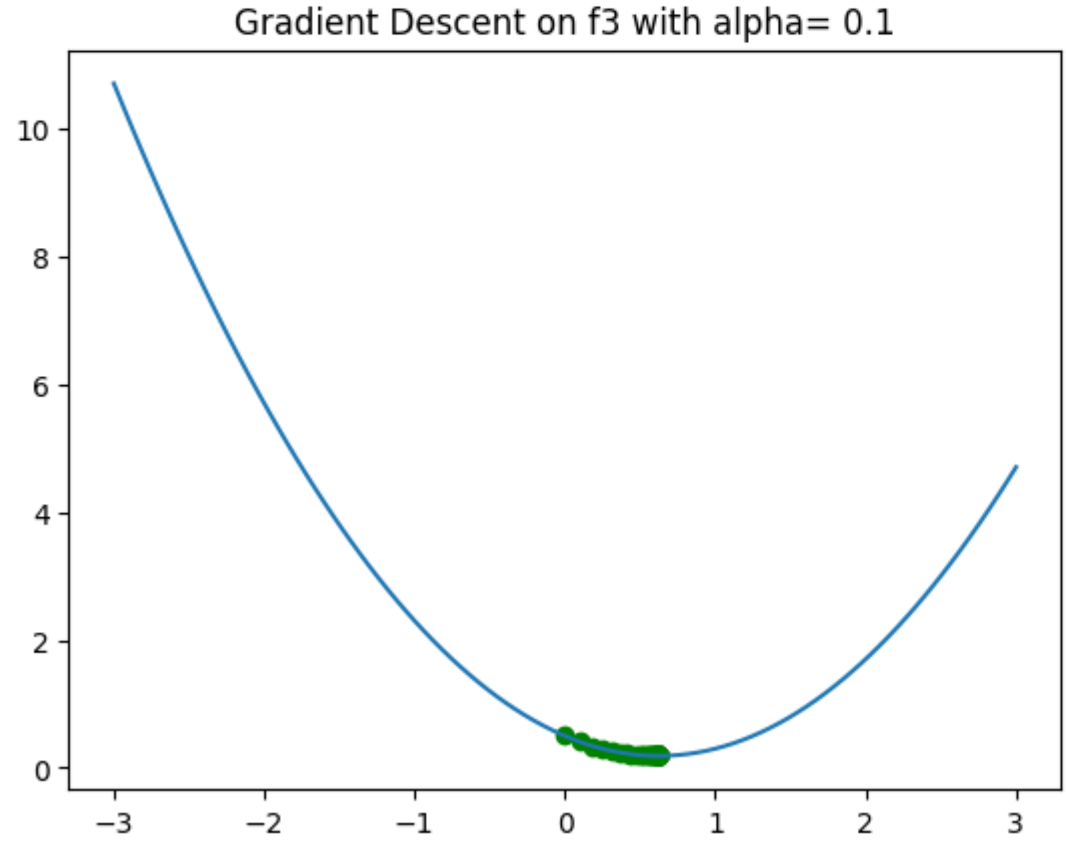
```
Minimum Found = [0.99415113 1.0036148 ] with 100 iterations
Minimum Found = [0.17566216 0.26128868] with 100 iterations
Minimum Found = [0.9999999  1.00000006] with 74 iterations
```



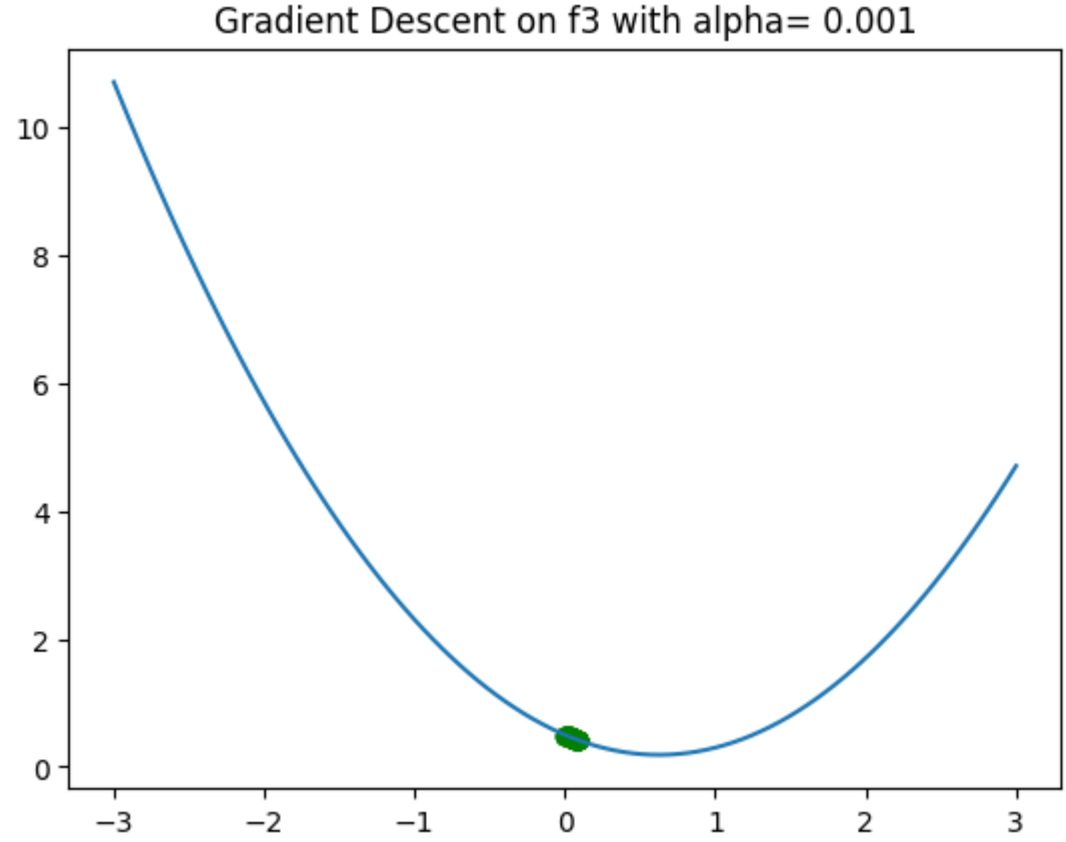


In [ ]:

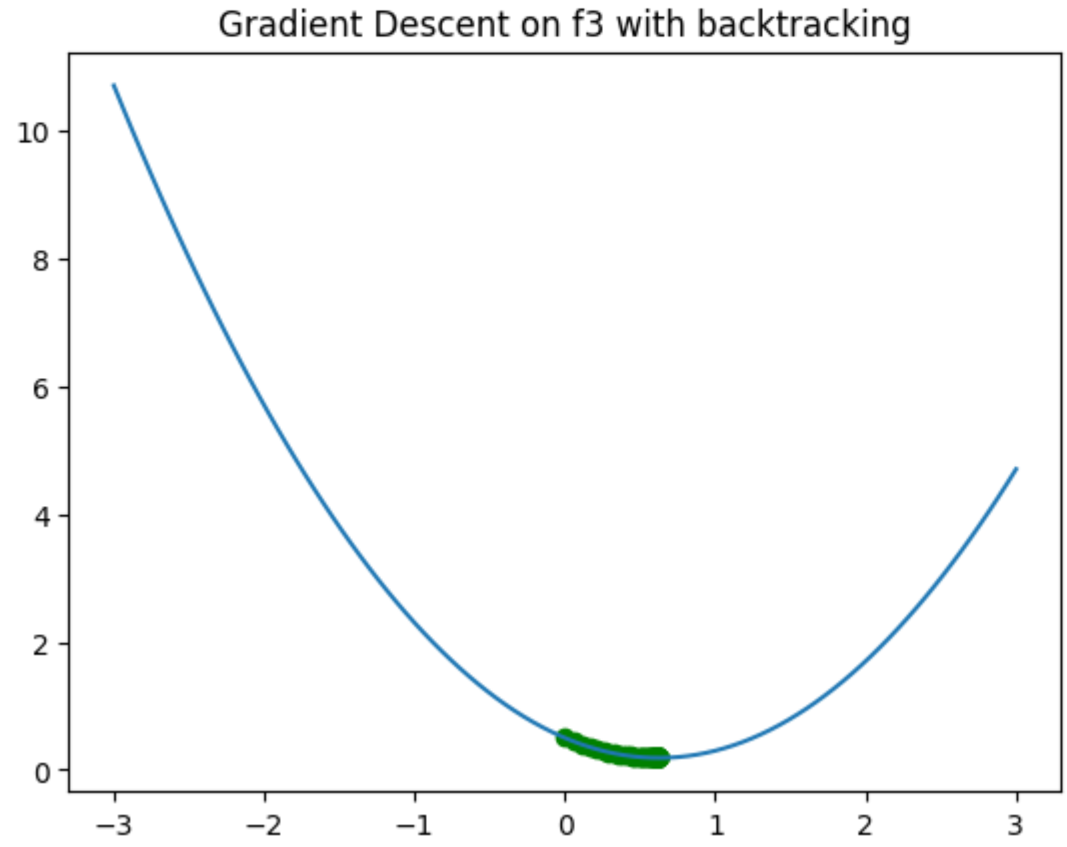
```
iters = []
err_vals = []
labels = []
err_xtrue = []
xtrue = np.ones(1).T

el1, el2, el3 = function_testing(f4, grad_f4, x0 = [0], title="Gradient D
iters.append(el1)
err_vals.append(el2)
labels.append("Alpha = 1e-1")
err_xtrue.append(el3)

el1, el2, el3 = function_testing(f4, grad_f4, x0 = [0], title="Gradient D
iters.append(el1)
err_vals.append(el2)
labels.append("Alpha = 1e-3")
err_xtrue.append(el3)

el1, el2, el3 = function_testing(f4, grad_f4, x0 = [0], title="Gradient D
iters.append(el1)
err_vals.append(el2)
labels.append("Backtracking")
err_xtrue.append(el3)

plot_error(iters, err_vals, labels)
plot_error(iters, err_xtrue, labels, title="Error (distance from x_true)"
```



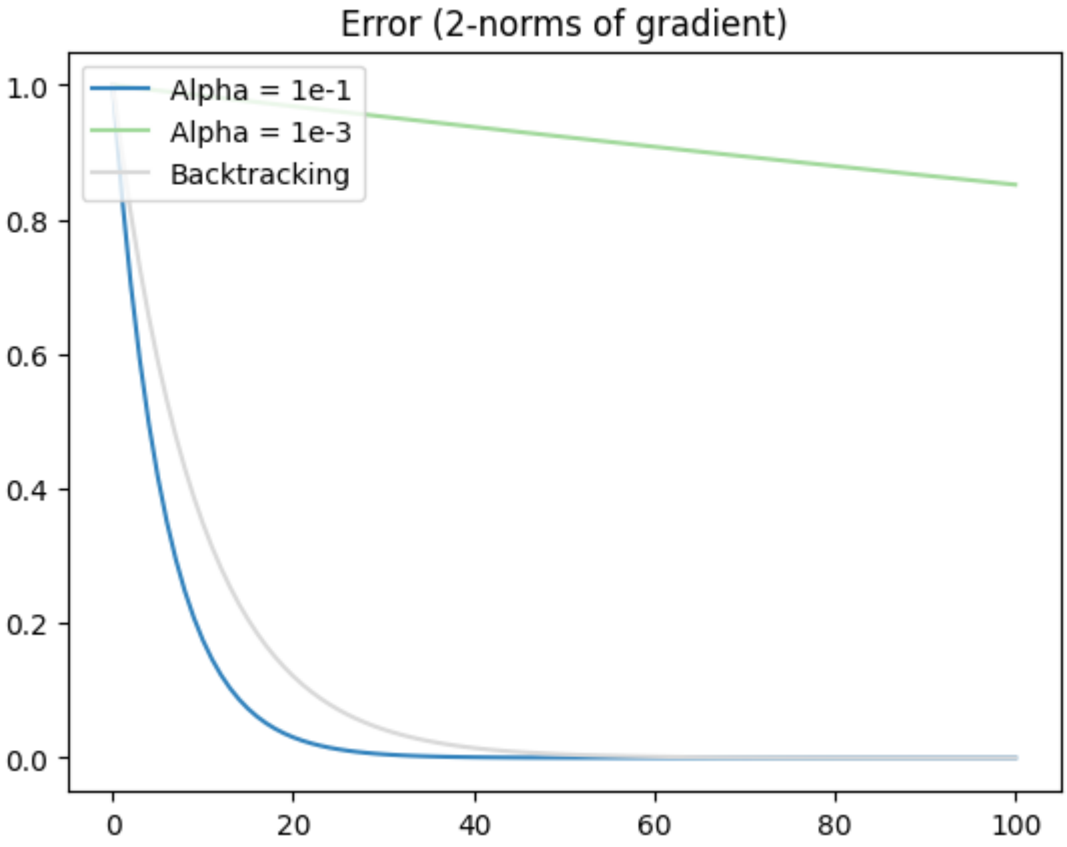
```
Minimum Found = [0.6247156] with 100 iterations
```

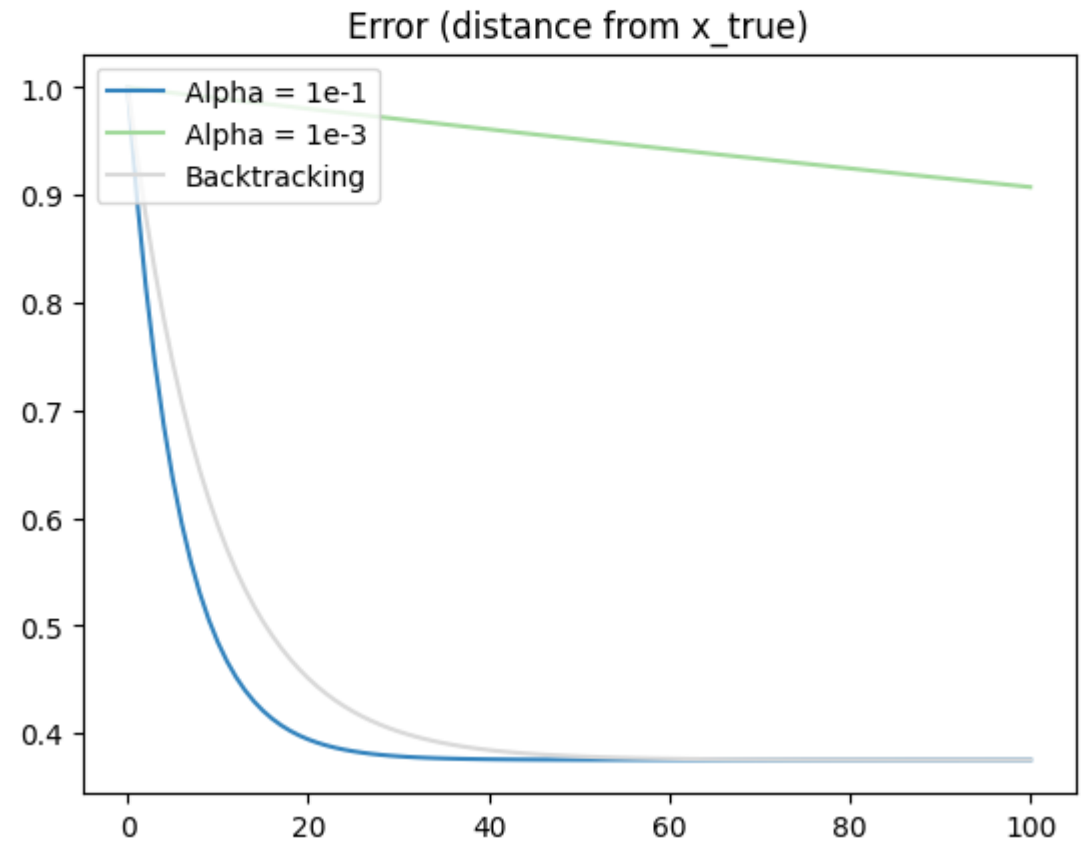


```
Minimum Found = [0.09162178] with 100 iterations
```



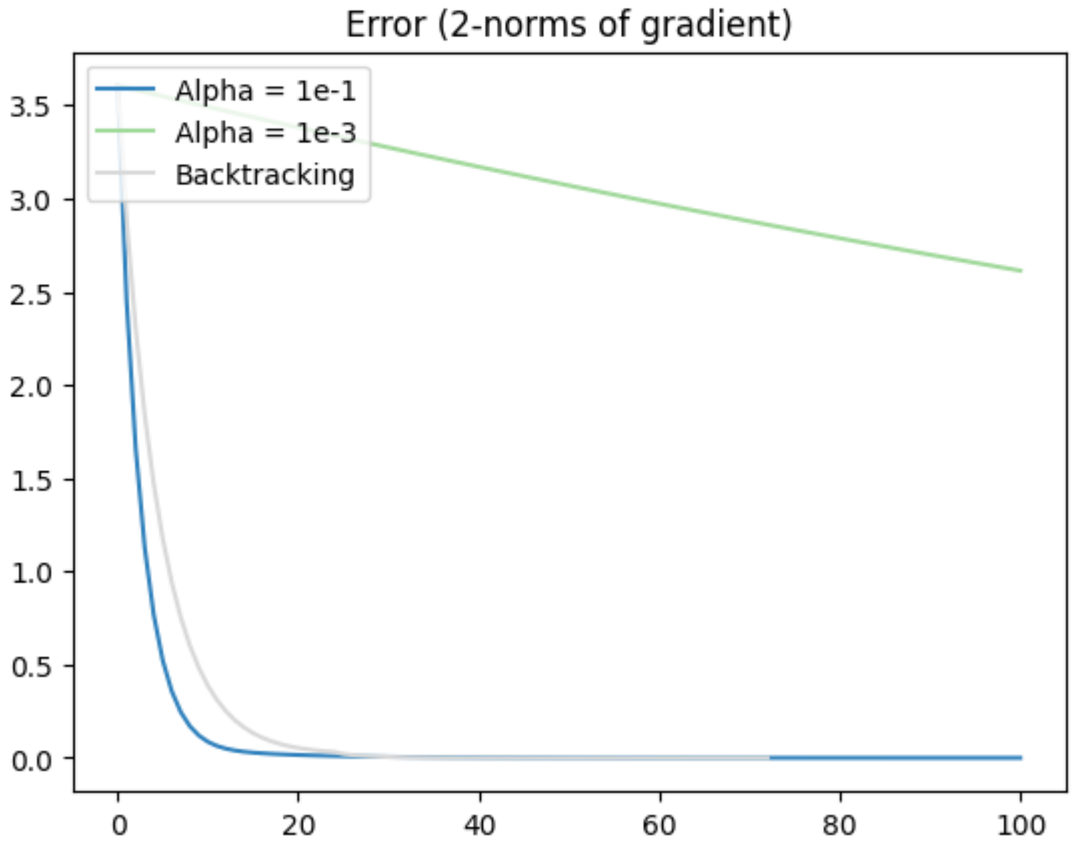
```
Minimum Found = [0.62469728] with 100 iterations
```

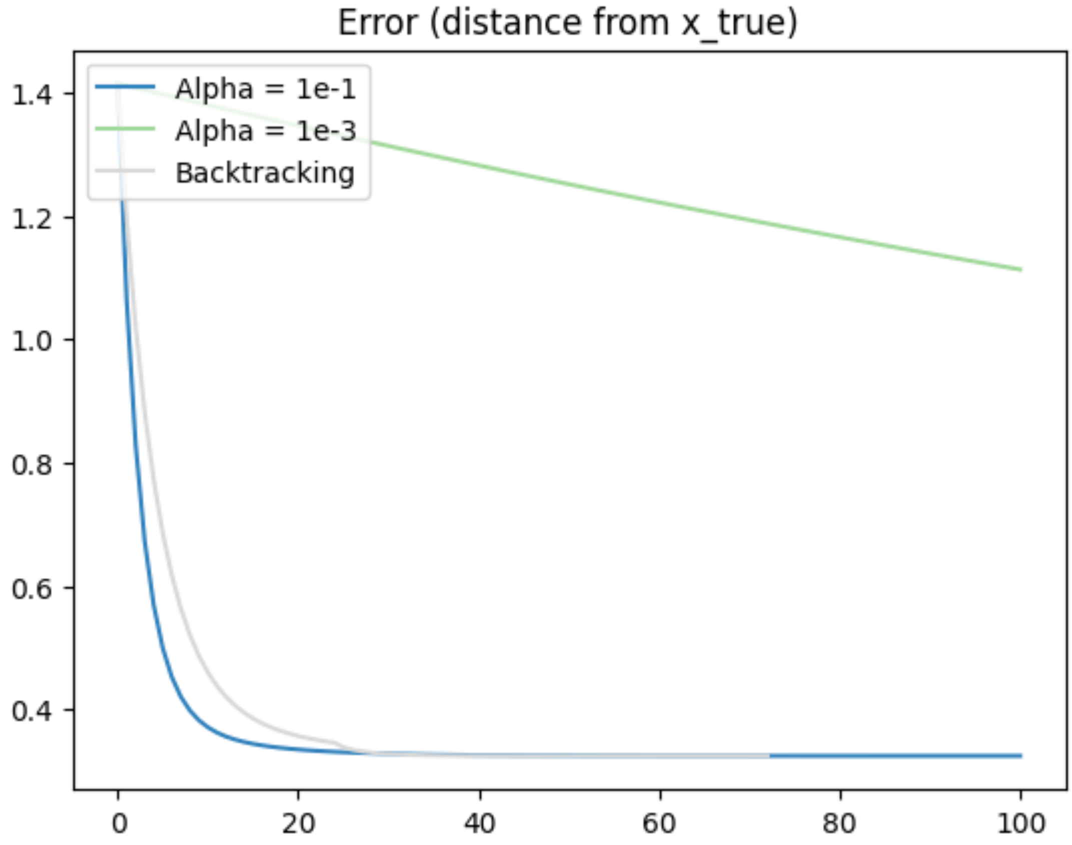




In [ ]:
```
iters = []
err_vals = []
labels = []
err_xtrue = []
xtrue = np.ones(2).T

el1, el2, el3 = function_testing(f4, grad_f4, x0 = [0, 0], title="Gradien
iters.append(el1)
err_vals.append(el2)
labels.append("Alpha = 1e-1")
err_xtrue.append(el3)

el1, el2, el3 = function_testing(f4, grad_f4, x0 = [0, 0], title="Gradien
iters.append(el1)
err_vals.append(el2)
labels.append("Alpha = 1e-3")
err_xtrue.append(el3)

el1, el2, el3 = function_testing(f4, grad_f4, x0 = [0, 0], title="Gradien
iters.append(el1)
err_vals.append(el2)
labels.append("Backtracking")
err_xtrue.append(el3)

plot_error(iters, err_vals, labels)
plot_error(iters, err_xtrue, labels, title="Error (distance from x_true)"
```

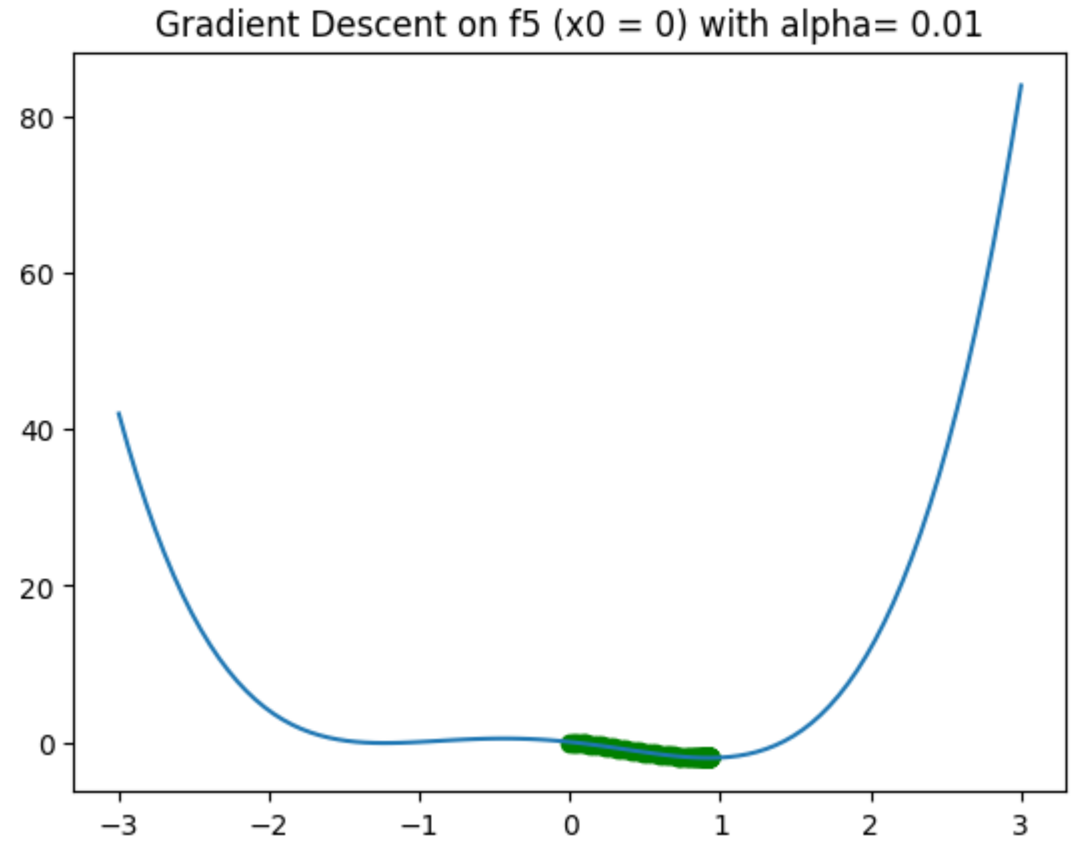
```
Minimum Found = [0.6959858  0.88591203] with 100 iterations
Minimum Found = [0.17078687 0.25405865] with 100 iterations
Minimum Found = [0.6959896  0.88590969] with 72 iterations
```


Error (2-norms of gradient)
Alpha = 1e-1
Alpha = 1e-3
Backtracking
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
20
40
60
80
100

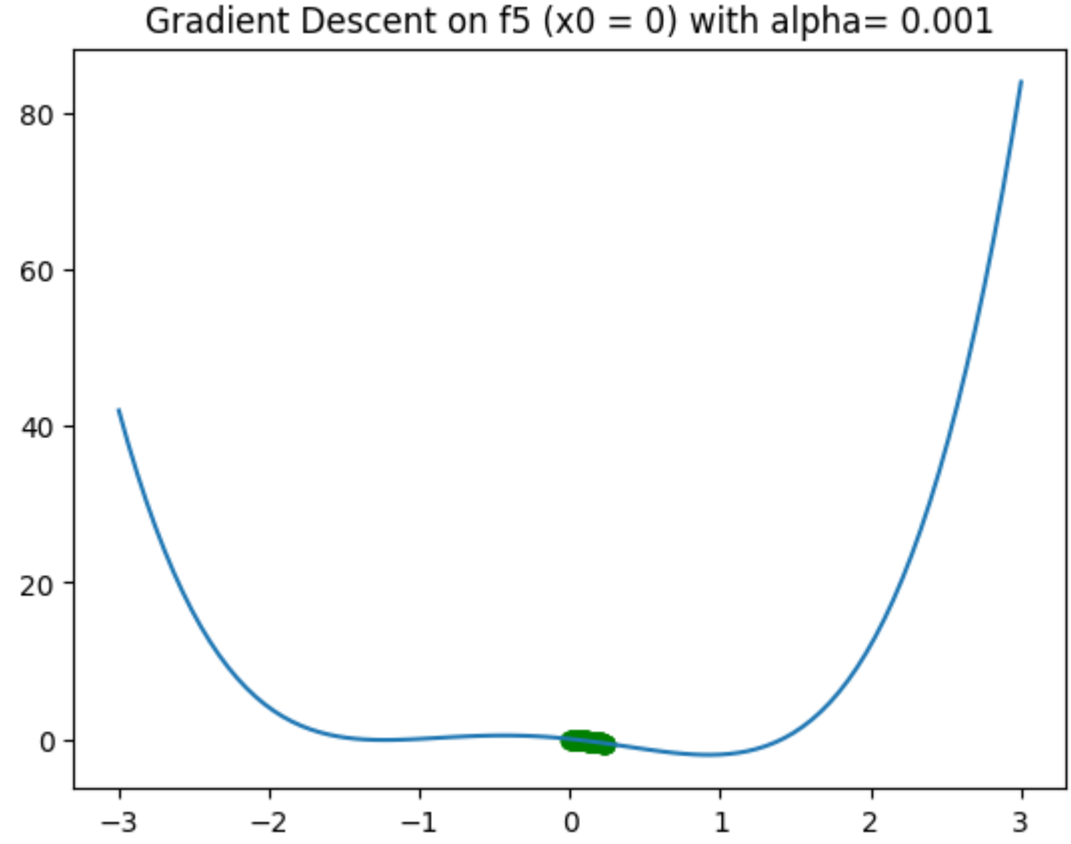

Error (distance from x_true)
Alpha = 1e-1
Alpha = 1e-3
Backtracking
1.4
1.2
1.0
0.8
0.6
0.4
0
20
40
60
80
100

In [ ]:

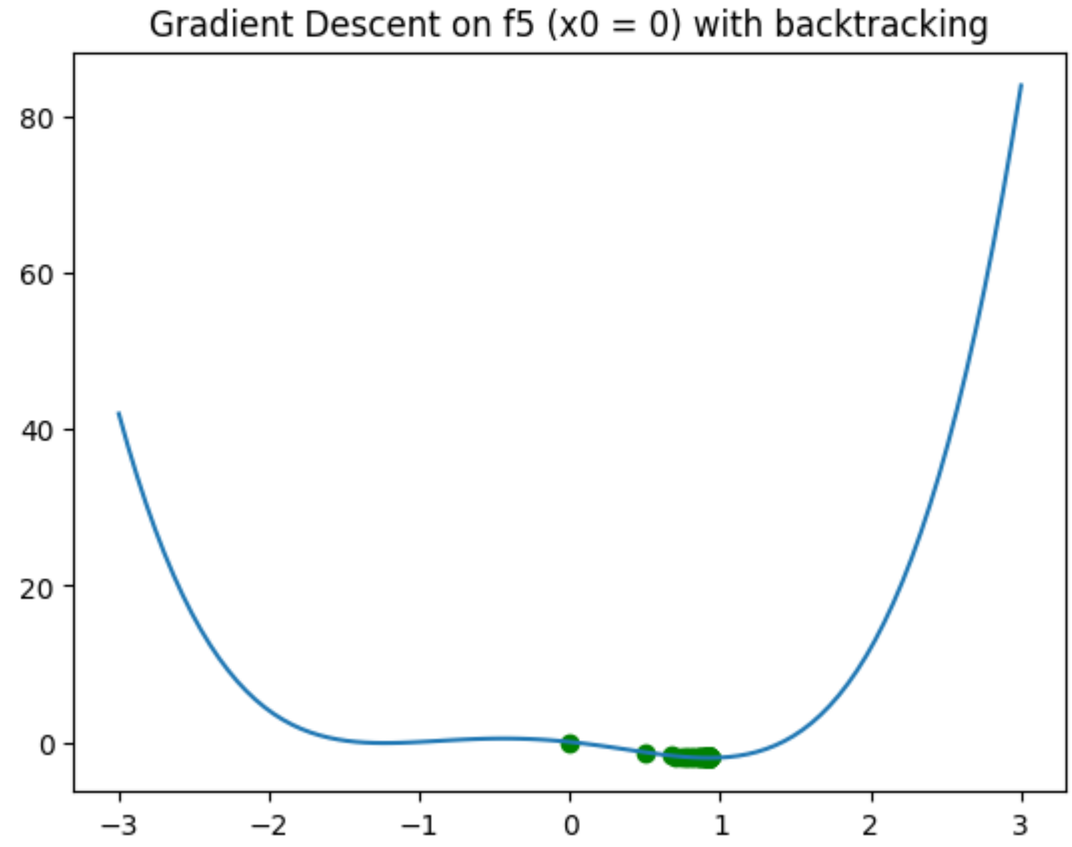
```
iters = []
err_vals = []
labels = []

el1, el2 = function_testing(f5, grad_f5, x0 = [0], title="Gradient Descen
iters.append(el1)
err_vals.append(el2)
labels.append("Alpha = 1e-1 (x0 = 0)")

el1, el2 = function_testing(f5, grad_f5, x0 = [0], title="Gradient Descen
iters.append(el1)
err_vals.append(el2)
labels.append("Alpha = 1e-3 (x0 = 0)")

el1, el2 = function_testing(f5, grad_f5, x0 = [0], title="Gradient Descen
iters.append(el1)
err_vals.append(el2)
labels.append("Backtracking (x0 = 0)")

plot_error(iters, err_vals, labels)
iters = []
err_vals = []
labels = []

el1, el2 = function_testing(f5, grad_f5, x0 = [-3], title="Gradient Desce
iters.append(el1)
err_vals.append(el2)
labels.append("Alpha = 1e-1 (x0 = -3)")

el1, el2 = function_testing(f5, grad_f5, x0 = [-3], title="Gradient Desce
iters.append(el1)
err_vals.append(el2)
labels.append("Alpha = 1e-3 (x0 = -3)")

el1, el2 = function_testing(f5, grad_f5, x0 = [-3], title="Gradient Desce
iters.append(el1)
err_vals.append(el2)
labels.append("Backtracking (x0 = -3)")

plot_error(iters, err_vals, labels)
```



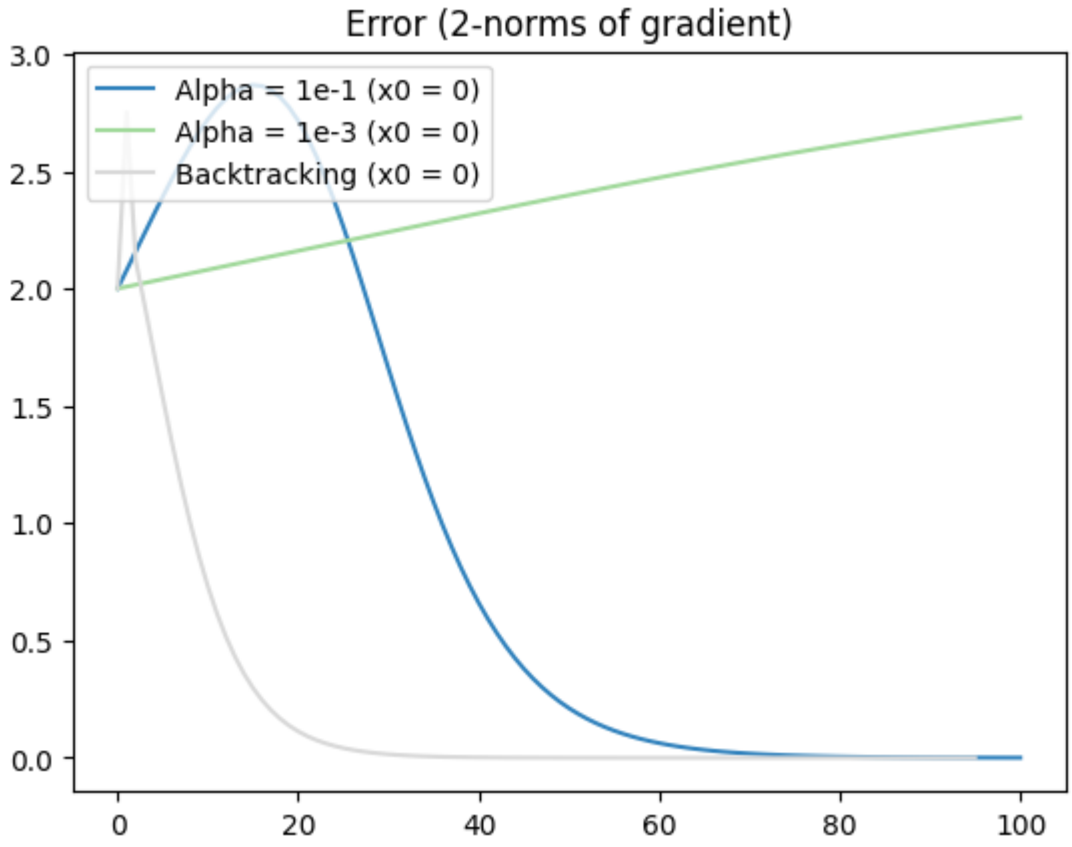
```
Minimum Found = [0.92218317] with 100 iterations
```

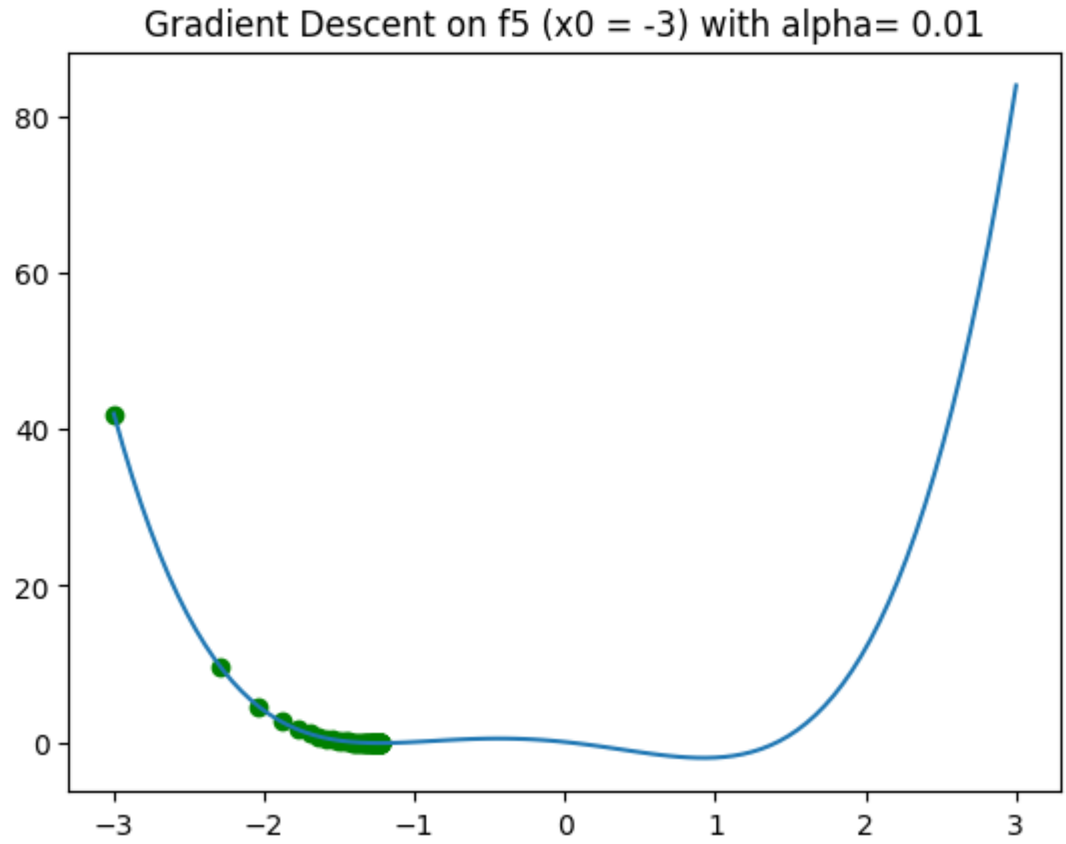


```
Minimum Found = [0.23560406] with 100 iterations
```

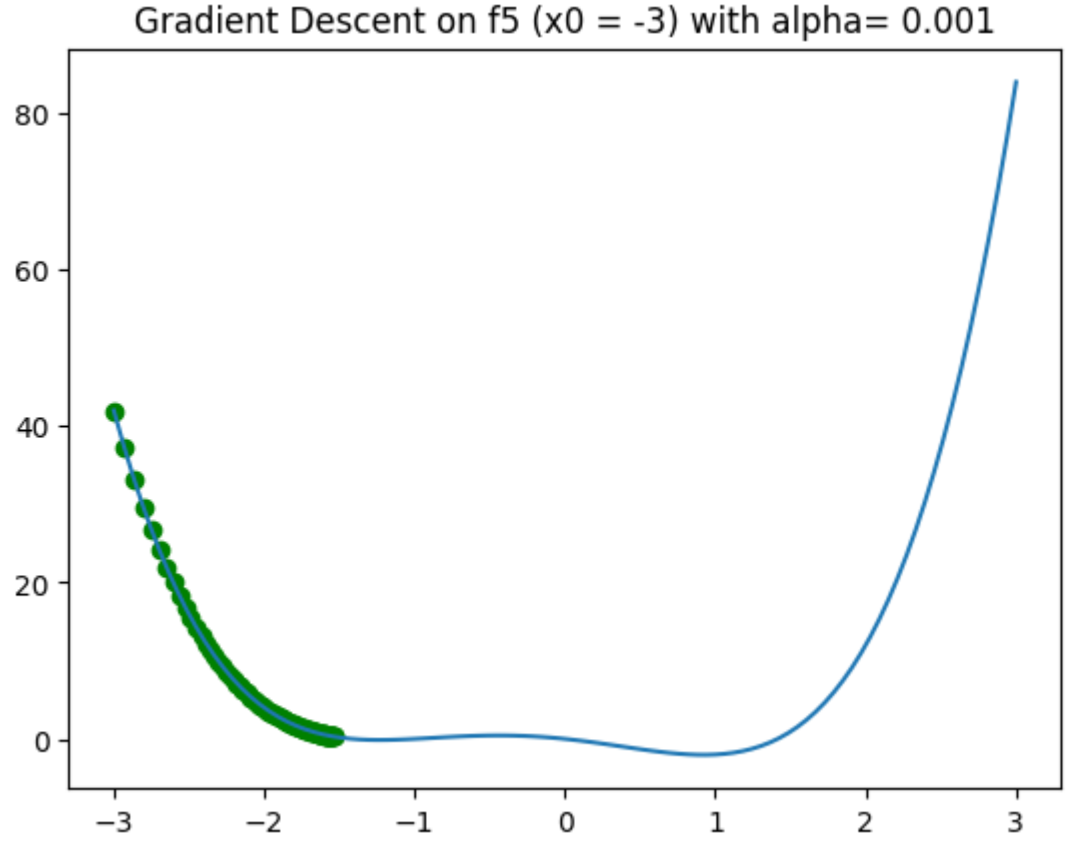


```
Minimum Found = [0.9222248] with 95 iterations
```

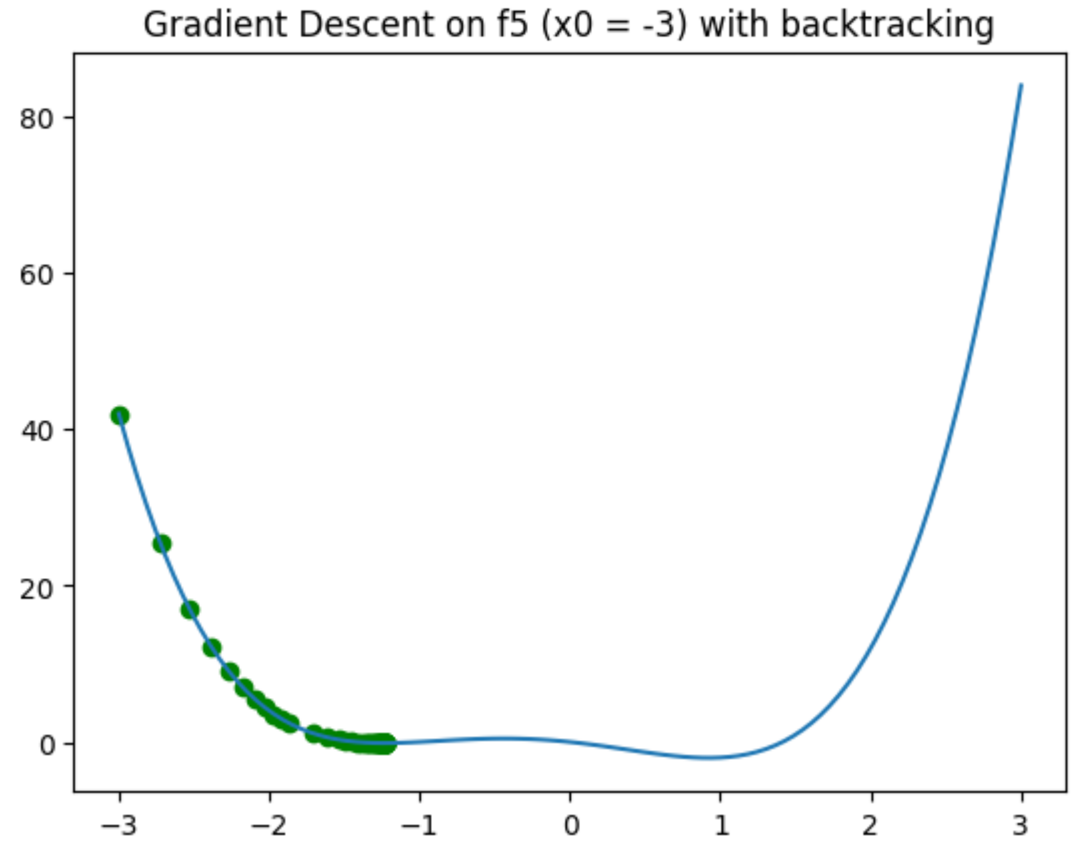




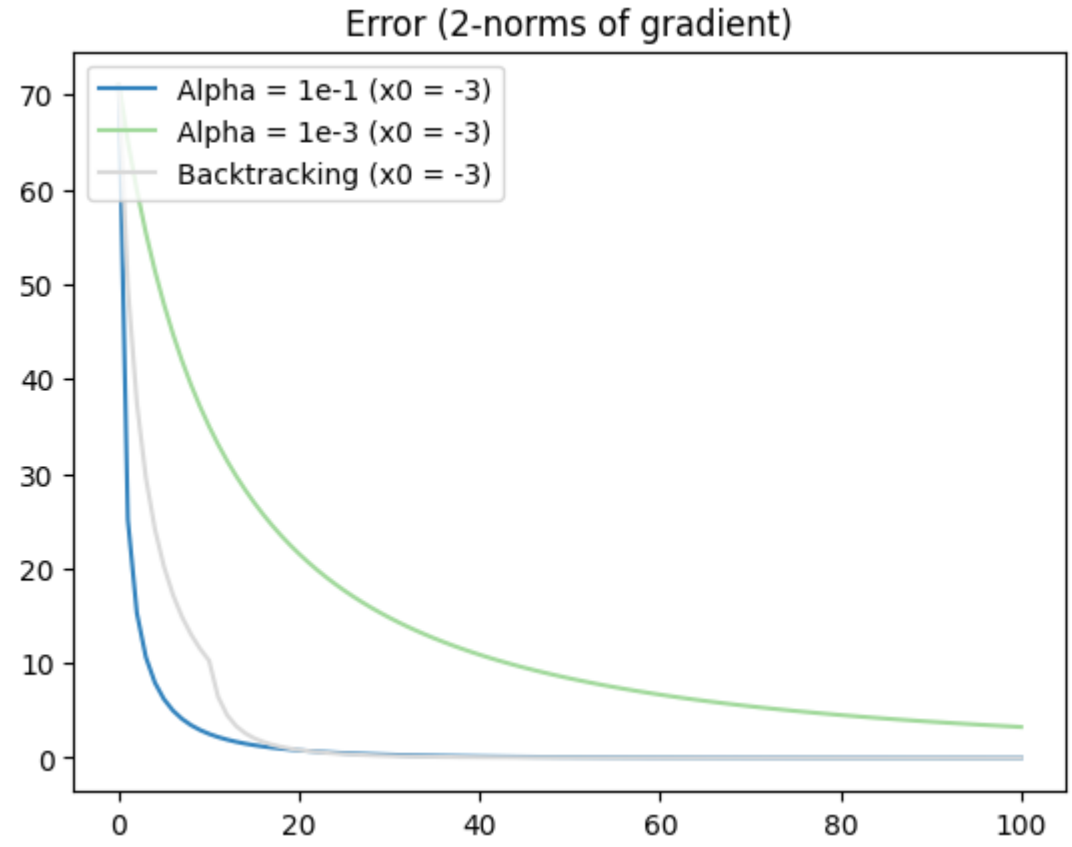
```
Minimum Found = [-1.23255127] with 100 iterations
```



```
Minimum Found = [-1.53828361] with 100 iterations
```



```
Minimum Found = [-1.23225111] with 100 iterations
```



STOCHASTIC GRADIENT DESCENT

```
In [ ]: # Utils

        def sigmoid(z):
            return (1 / (1 + np.exp(-z)))

        def f(w, xhat):
            return sigmoid(xhat.T @ w)

        def grad_f(w, xhat):
            return (sigmoid(xhat.T @ w) * (1 - sigmoid(xhat.T @ w)) * xhat.T)

        def MSE(f_w_x, y):
            return np.linalg.norm((f_w_x-y))**2

        def grad_MSE(grad_f_w_x, f_w_x, y):
            return grad_f_w_x.T * (f_w_x - y)

        def ell(w, X, Y):
            d, N = X.shape

            mse_sum = 0
            for i in range(0, N):
                mse_sum+=MSE(f(w, X[:, i]), Y[i])

            return mse_sum / N

        def grad_ell(w, X, Y):
            d, N = X.shape

            grad_mse_sum = 0
            for i in range(0, N):
                grad_mse_sum += grad_MSE(np.array(grad_f(w, X[:, i])), f(w, X[:,

            return grad_mse_sum / N
```

In [ ]:

```
def SGD(l, grad_l, w0, D, batch_size, n_epochs, tolf=1e-12, tolx=1e-12, k
    alpha = 1

    # D = (X, Y) where X is d x N
    #                  Y is N
    X, Y = D
    X_backup = X
    Y_backup = Y
    d, N = X.shape

    n_batch_per_epoch = N // batch_size
    w_vals = [w0]
    f_vals = [ell(w0, X, Y)]
    grad_f_vals = [grad_ell(w0, X, Y)]
    err_vals = [np.linalg.norm(grad_ell(w0, X, Y))]

    k= 0
    for epoch in range(n_epochs):
        idx = np.arange(N)
        np.random.shuffle(idx)

        for k in range(n_batch_per_epoch):
            batch_indices = idx[k * batch_size : (k + 1) * batch_size]

            other_1 = list(idx[:k * batch_size])
            other_2 = list(idx[(k + 1) * batch_size:])
            other_indices = np.array(other_1 + other_2)
            # Sample M from D
            Mx = X[:, batch_indices] # Has shape d x batch_size
            My = Y[batch_indices] # Has shape batch_size
            M = (Mx, My)

            # Remove Mx and My from X and Y
            # X = X[:, other_indices]
            # Y = Y[other_indices]

            # Update w
            w = w0 - alpha * grad_ell(w0, Mx, My)
            w_vals.append(w)
            # Restart
            w0 = w

        # Reload X and Y
        X = X_backup
        Y = Y_backup
        f_vals.append(ell(w, X, Y))
        grad_f_vals.append(grad_ell(w, X, Y))
        err_vals.append(np.linalg.norm(grad_ell(w0, X, Y)))
        ## ATTENTION: You have to shuffle again (differently)

    return w, f_vals, grad_f_vals, err_vals

# Remember: In SGD, w0 SHOULD be chosen randomly (sample from Gaussian)
```

In [ ]:

```
def x_split(X, Y, N_train):
    d, N = X.shape

    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

def get_digits(X, Y, chosen_numbers):
    I = []
    X_def = []
    Y_def = []

    d, N = X.shape
    for number in chosen_numbers:
        I.append((Y == number))

    for num_indeces in I:
        X_def.append(X[:, num_indeces])
        Y_def.append(Y[num_indeces])

    return X_def, Y_def, np.concatenate(X_def, axis=1), np.concatenate(Y_
```

In [ ]:

```
import pandas as pd
```

In [ ]:

```
data = pd.read_csv("./data.csv")
data = np.array(data)
```

In [ ]:

```
X = data[:, 1:].T
Y = data[:, 0]

chosen_digits = []
chosen_digits.append(int(input("Insert first number")))
chosen_digits.append(int(input("Insert second number")))
```

In [ ]:

```
X_set, Y_set, X_cat, Y_cat = get_digits(X, Y, chosen_digits)
```

In [ ]:

```
N_train = 30000
X_train = []
X_test = []
Y_train = []
Y_test = []

for my_set in zip(X_set, Y_set):
    Xtrain, Xtest, Ytrain, Ytest = x_split(my_set[0], my_set[1], N_train)
    X_train.append(Xtrain)
    X_test.append(Xtest)
    Y_train.append(Ytrain)
    Y_test.append(Ytest)

X_cat_train, X_cat_test, Y_cat_train, Y_cat_test = x_split(X_cat, Y_cat,

D = (X_cat_train, Y_cat_train)
```

In [ ]:

```
d, N = X_cat_train.shape
w0 = np.random.normal(0, 0.1, d)
batch_size = 5
n_epochs = 10
w, f_vals, grad_vals, err_vals = SGD(ell, grad_ell, w0, D, batch_size, n_
```

In [ ]:

```
# plt.plot(np.arange(d), w)
# plt.show()
# plt.plot(np.arange(11), np.array(grad_vals))
# plt.show()
print(w)
```

```
[ 6.07643390e-02 -6.74054270e-02 -5.23109864e-02  1.76363703e-01
  1.30476045e-01 -2.00297542e-01  2.67028363e-02  1.06233776e-01
 -5.03875930e-02 -3.33099042e-02 -7.24775749e-02  1.01180143e-01
 -2.23887502e-01 -8.73384556e-02  4.58489369e-03  1.40280041e-01
  6.54276498e-02  1.47111406e-01 -4.38880470e-02 -8.25666537e-02
  1.01153026e-01 -1.43961629e-01 -1.16964079e-01 -7.57430199e-02
  6.57550338e-02  2.96242135e-02 -1.04245150e-01  1.74832180e-01
  5.77277275e-02  5.79150952e-02 -3.79300442e-02  7.13882419e-02
  7.64038743e-02 -2.42561735e-03 -8.48814980e-02  9.34362952e-03
 -6.96232520e-02 -1.34231089e-01  1.49124315e-01  4.14136906e-03
 -1.35996377e-01  1.44354796e-02  1.29984072e-01 -2.20371621e-02
 -9.06875092e-02  5.10782687e-02  4.47807560e-02 -7.45111148e-02
 -1.20371209e-01  5.25143614e-02  5.25134188e-02 -3.32469846e-02
  1.36263962e-02  2.71518272e-02 -3.32094053e-02  1.18176859e-01
 -7.05916791e-02  1.15161609e-01 -3.82232878e-02 -6.08149175e-02
  6.39963016e-02  1.09887726e-01  9.22567778e-03 -5.37682003e-02
  1.28251390e-01  1.67445094e-01  1.78432297e-02  1.20789490e-01
 -1.67910057e-01  5.85548101e-02  3.87646880e-02  1.96059175e-01
 -2.17423810e-01 -3.84930983e-02 -1.17059873e-01  6.94627219e-02
  7.49766778e-02  4.00944844e-02  1.94317747e-01 -7.01289105e-02
 -1.99747117e-02  5.23831958e-02  7.62379276e-02 -7.59994388e-02
  1.25148139e-02  1.48486156e-01 -4.89702806e-03  5.94891232e-02
  1.08837171e-01 -1.78396172e-01 -4.95842689e-02  1.46624853e-01
 -1.34230617e-01 -1.65595734e-01  1.92617882e-01 -1.46611116e-02
  3.17128879e-02  1.82508243e-01  1.46555966e-01 -2.21735769e-01
  1.56598171e-01  9.16933389e-03  1.63033100e-01  1.22367950e+00
  2.86931993e+00  4.53457172e+00  4.48086133e+00  1.78895327e+00
  6.39831709e-04  6.96724258e-02 -2.12539044e-01  1.09829760e-01
 -3.01339035e-02 -4.41345932e-02 -8.63003202e-02  2.50373049e-02
 -8.24298120e-02  2.55543866e-02  1.19390147e-01 -8.13538710e-02
 -1.22989095e-01 -7.91776523e-02 -2.91752465e-02 -8.02077957e-02
  5.40134549e-03 -3.47876771e-02  1.46884416e-02  5.33196702e-02
  5.09158071e-01  2.96321543e+00  4.05266245e+00  4.44713335e+00
  4.31633735e+00  4.50828124e+00  4.33685437e+00  2.64479369e+00
  1.26417121e-01  9.79687447e-02 -4.00003921e-02 -8.17930855e-02
 -8.09840932e-02  7.31644453e-02  2.18182740e-02  5.96829269e-02
 -1.40074603e-01 -1.05628656e-01 -2.22953026e-02  5.65725805e-03
 -1.61633528e-01  4.36222319e-03 -1.86708127e-01  7.55097219e-02
  5.51208291e-02 -2.96352773e-02  2.21526942e-02  9.27373593e-01
  3.62598135e+00  4.47502806e+00  4.38160310e+00  4.38721036e+00
  4.52886643e+00  4.45951230e+00  3.06788327e+00  3.87178275e-01
 -1.42282566e-01  1.27203297e-01 -1.47550119e-01 -2.52750017e-01
  1.33264844e-01  2.42102877e-02  1.55493410e-01 -3.85487601e-02
 -4.14958950e-02  2.72369687e-02  4.41353733e-02  8.30384735e-02
 -8.27735043e-02 -8.57405210e-02 -1.90338396e-01  1.41398590e-01
 -8.71151095e-02  1.42848561e-02  1.81851516e+00  3.72053063e+00
  4.58856213e+00  4.62224865e+00  4.57347588e+00  4.38855664e+00
  3.81506641e+00  1.13582725e+00  1.84329684e-01 -6.69954548e-03
 -1.71647957e-01 -1.33951268e-01 -1.68729302e-02 -1.46198365e-01
  4.97366691e-02  1.86812205e-02 -5.16930944e-02  1.91055495e-02
 -1.37837279e-01 -1.16450579e-02  1.00136146e-01  9.40156985e-03
  3.71012521e-02 -1.06351480e-02  1.59818588e-01  1.02779489e-01
  1.06528875e+00  2.88061951e+00  4.20671866e+00  4.47030742e+00
  4.32154007e+00  4.36947444e+00  4.51651143e+00  3.17167695e+00
  5.84034727e-01 -6.56900044e-02 -3.89749478e-02  7.23970710e-02
  2.32226415e-02 -7.41073877e-02 -6.37789962e-02 -1.33592878e-01
  8.47428182e-02 -1.00249943e-01 -1.47304436e-02 -2.72599553e-02
  8.23735947e-02  2.61897169e-01  4.36512121e-02  4.61150816e-02
  1.42052591e-03 -1.06896184e-01  1.27881690e+00  3.74870758e+00
  4.33373038e+00  4.51466397e+00  3.95212394e+00  2.80131960e+00
  1.19823969e+00  1.87121562e+00  1.13294065e+00  2.36482351e-02
 -2.50852598e-01 -1.74342810e-01  4.67542393e-02 -5.40338101e-02
  1.88281067e-03  3.21116108e-02  1.75656885e-01 -1.83239271e-02
 -2.76208427e-01  9.93591832e-03 -6.69080731e-02  6.20000094e-02
  9.54472424e-02  2.57808864e-02 -8.56000169e-02  1.27145846e-01
 -1.00869392e-01  1.88923430e+00  4.32474376e+00  4.33417070e+00
  4.30181895e+00  3.03731688e+00  6.90728097e-01  9.16493433e-02
  7.92603000e-02 -1.62951955e-02  6.85061908e-02 -3.35585854e-02
 -2.67854432e-02 -2.81502974e-02  4.64560419e-02  1.29321821e-01
 -4.77734581e-02 -6.92002235e-02  1.44268092e-02 -3.23709084e-02
 -1.09107936e-02 -2.61146871e-01  6.99300014e-02  2.78876933e-02
  8.16690509e-02 -6.34158742e-02  1.14115380e-01  1.82678056e-01
  1.79416448e+00  4.34212878e+00  4.36712686e+00  4.35363187e+00
  1.49793496e+00  1.55555201e-01 -1.12053987e-01 -2.57398525e-02
 -4.79129235e-02 -1.43591020e-01 -4.86721547e-03 -1.77015012e-01
  1.68827593e-01  2.47324763e-02  9.62123547e-02 -1.66118433e-01
 -1.38794839e-01  5.57665303e-02 -7.23359567e-02 -2.49562949e-01
 -1.71423741e-01 -5.58469902e-03  2.84494705e-02  6.70127412e-03
  2.34275687e-01  1.39265760e-01 -1.37385426e-01  8.34806549e-01
  4.46647205e+00  4.67350875e+00  4.27589194e+00  5.69073098e-01
 -1.80032642e-02 -5.60854811e-02 -1.24833789e-01  5.67753862e-02
  4.68992804e-02 -1.49402179e-01 -3.69392470e-02 -9.11499892e-02
 -3.82198880e-02 -1.00829359e-01  3.68329567e-02 -3.18110392e-03
  8.20532336e-03 -1.54266712e-01  1.11463651e-01  8.90991692e-03
  4.01507840e-02  4.45440496e-02 -1.62079324e-01 -3.38432306e-02
  7.24632173e-02  2.08235448e-02 -1.82363408e-02  2.91239367e+00
  4.51855414e+00  4.41894402e+00  1.90427117e+00 -3.53110382e-03
  2.59566534e-03  8.58561508e-02 -5.45386751e-02 -1.57196858e-01
  9.28627785e-02  1.36134038e-02  1.39637361e-01  1.17245582e-01
  5.46979201e-03  1.28504262e-01 -3.94372119e-02 -6.06483530e-02
  1.45600265e-01  3.62091481e-02  4.36644021e-02 -5.87924129e-02
  4.56785655e-02  2.37180972e-02  2.10652224e-02  8.31424256e-02
  8.31175926e-02 -4.30720355e-02  1.03319542e+00  4.46335570e+00
  4.47046654e+00  4.35545022e+00  4.19578425e-01  1.20774074e-02
  2.23498940e-02 -7.57944407e-02  7.31997489e-02 -2.52185058e-01
 -1.62108889e-01 -1.69941501e-01  1.01681251e-01  1.17344184e-01
  9.26012265e-02  2.47613497e-03 -3.55420240e-02  1.32496882e-01
  4.48490474e-02 -3.56215301e-02 -1.12523945e-01 -8.83497300e-02
 -9.95095935e-03 -1.55665707e-01  3.69556769e-02  3.98243251e-02
  1.40962888e-01  1.38917972e-01  3.62949039e+00  4.40238685e+00
  4.41118527e+00  2.81044510e+00  1.82603304e-01 -2.00401036e-02
  1.06821464e-01 -7.44729234e-03 -1.39047026e-02 -2.79046285e-02
  1.78135753e-01  1.64307819e+00  6.52951168e-01  1.50533477e-01
 -4.16408895e-02 -1.09164263e-01  1.93007987e-01 -2.32562984e-02
  5.03674440e-02  6.49890285e-02  6.43500913e-03 -5.32415742e-02
  1.21548930e-02  1.25349086e-01 -1.76049644e-01 -1.70519309e-02
 -1.32913637e-01 -2.25856200e-02  3.93823233e+00  4.27840312e+00
  4.42374317e+00  3.93104161e-02 -9.48753110e-02 -1.97808514e-02
 -1.20370387e-01 -4.46691532e-02  7.21013081e-01  2.69289371e+00
  4.40779034e+00  4.36312337e+00  4.39861439e+00  3.07169037e+00
  9.68009912e-01  9.76930107e-02  3.75572837e-02 -1.91973466e-02
 -4.02098391e-02  1.06591293e-01 -3.95963292e-02 -1.03728208e-01
 -5.88605810e-02 -2.35876006e-02  7.13012602e-02  1.00119074e-01
 -1.73469279e-01 -1.84803851e-02  2.37520072e+00  4.31387273e+00
  4.41310648e+00  8.72776647e-02 -2.44960233e-02 -1.17485963e-01
  3.66451990e-02  8.18016314e-01  4.07775795e+00  4.38058425e+00
  4.35500123e+00  4.52745961e+00  4.34314969e+00  4.70158608e+00
  4.17112538e+00  2.50623941e-02  1.46207126e-01  9.61925923e-02
  3.97184395e-02 -1.51713275e-01  8.77642269e-02 -1.13384198e-01
  6.19041471e-02 -2.97522648e-02  1.92585865e-01  5.21991473e-02
  1.96291020e-01  6.77795386e-02  1.18278488e+00  4.36192421e+00
  4.47971114e+00  9.39857915e-01  7.88034437e-02 -5.45294049e-02
  7.49590399e-01  4.18047327e+00  4.38791123e+00  4.20944752e+00
  2.22676882e+00  7.67881255e-01  1.33384997e+00  3.87140978e+00
  4.48928430e+00  2.17431988e+00 -2.86522816e-01  6.40958554e-02
  1.37548451e-01  1.28812791e-01  4.43298820e-02  6.96581115e-02
 -7.42701371e-02 -4.12497779e-02 -1.61267224e-01  1.50451196e-01
 -9.44546510e-02  2.02383489e-01  8.10490091e-01  4.55085010e+00
  4.54444899e+00  4.06855476e+00  4.38182635e-01  2.14086067e+00
  4.07268881e+00  4.41387876e+00  3.48859263e+00  9.14637474e-01
  5.33057764e-02 -1.46346555e-01 -1.02594906e-01  1.44670058e+00
  4.28365351e+00  3.85854103e+00  1.09518696e-01 -2.48572713e-01
 -1.10554814e-01 -1.52719186e-02 -5.94112929e-02  1.66293410e-01
  1.25473036e-01 -3.36643529e-02  9.47995876e-02  7.22522104e-03
 -1.26551252e-01 -8.92802980e-02  4.64936340e-02  2.52062366e+00
  4.57056400e+00  4.34756290e+00  4.44052856e+00  4.56264454e+00
  4.33182867e+00  2.23879736e+00  5.73567994e-02  1.40918841e-01
 -9.65510626e-03 -8.83427696e-03  4.74644251e-02  8.96229857e-01
  4.46801512e+00  3.57020474e+00  1.97207976e-01  8.87894149e-02
  5.39775608e-02  1.14351582e-01 -3.13016341e-02  8.77359003e-02
  5.16646952e-02  5.20738111e-02  4.79735751e-02 -1.35302750e-01
  7.65888110e-02 -3.22365608e-02  1.20586630e-02  4.37487698e-01
  4.55556194e+00  4.47215695e+00  4.46603281e+00  4.25634480e+00
  2.60683729e+00  1.47712657e-01  8.07389725e-02  8.37931439e-01
  8.68260921e-01  8.73802181e-01  2.64055103e+00  4.16143137e+00
  4.28801655e+00  1.28817276e+00  3.36461515e-02  2.91884636e-01
  1.13200171e-01 -9.20554349e-03 -7.44048983e-02  5.49352609e-02
 -3.72420042e-02  1.11529244e-01 -1.12039373e-01 -6.42368048e-02
 -1.30214014e-01  5.78414982e-02  5.96444166e-02  5.59635017e-01
  4.44275787e+00  4.30854212e+00  4.45566198e+00  4.39664111e+00
  4.32952010e+00  3.43610226e+00  3.83788858e+00  4.44453867e+00
  4.38409899e+00  4.18779893e+00  3.86175166e+00  3.26478017e+00
  2.49696114e+00  1.24178398e-01  1.57766606e-01 -1.05367747e-01
  1.45264843e-01  2.90728132e-02  1.19797428e-01 -1.07855685e-01
 -5.64496256e-02 -6.60587776e-02 -4.64745079e-02  6.83452272e-02
  4.84137914e-02 -3.16819028e-02  2.16866772e-01  6.90902852e-01
  3.71609577e+00  1.68861446e+00  2.67217495e+00  4.22405323e+00
  3.32329714e+00  3.01282244e+00  2.95505981e+00  2.22789558e+00
  1.08967269e+00  4.08230315e-01  1.25021327e-01  1.40004896e-01
 -1.78954581e-01  1.03470716e-01  1.41588324e-01 -4.70435239e-02
  2.49728850e-02 -8.48862647e-02  6.42577570e-03  9.45368503e-02
  4.45877171e-03  3.38126725e-01  7.29057229e-02  5.19185748e-02
 -9.16606075e-02 -1.02210788e-01 -3.22974910e-03 -6.84953288e-02
  4.53978511e-02  8.89357489e-02 -1.88360342e-02  2.85018215e-02
  8.53646824e-02 -1.04150160e-01  1.92567460e-01 -3.42703836e-02
  1.65488915e-01 -7.46212560e-02  9.67428412e-02  6.64847463e-02
  5.77752591e-02  8.91040468e-02  1.16255156e-01  9.86738272e-02
  1.32369792e-02 -4.42442960e-02 -3.85306881e-02 -3.45654700e-02
 -1.53180426e-01 -1.37586694e-01 -3.49894563e-02  2.86959929e-02
  5.39594716e-02 -3.25826196e-02  1.59875097e-02  9.15198611e-02
  1.50023876e-01  3.56699545e-02  1.34642140e-01  1.20119492e-01
  6.43580582e-02  7.16726272e-02 -4.31628726e-02 -1.55419629e-01
  4.52669758e-02 -1.84063310e-01 -1.50822866e-01  8.96648732e-02
 -4.16221561e-02 -2.50609518e-01 -7.74541444e-02 -3.99802337e-02
 -5.37361286e-02  6.83644677e-02  2.15891748e-01 -2.08906300e-01
  7.01346429e-02  1.48366407e-01  1.67797609e-01 -7.19515263e-02
 -2.79659492e-02 -6.92550340e-02  1.09343744e-01 -7.58558072e-02
 -3.01408219e-02 -7.19561009e-02  1.55571756e-02  1.92653501e-01
 -6.44827759e-02 -1.97628359e-01 -9.43915422e-02 -5.54495476e-02
  4.47058359e-02  9.17722866e-02 -1.00605192e-01 -9.95475739e-02
 -2.17172668e-03  1.51404091e-01 -7.15953329e-02  1.69459734e-01
 -1.05030179e-01 -7.37002059e-02 -2.42989747e-02 -6.96420198e-02
 -2.56051577e-01  1.26235950e-01  1.57169687e-01 -7.57993954e-02
  8.58366297e-02 -9.85168547e-02  5.92976155e-04 -1.48139872e-02
 -2.29424734e-03 -1.82905277e-02 -7.63745274e-02 -4.21155278e-02
  8.71986860e-02 -1.03985597e-02 -4.28682610e-02 -2.09629772e-01
 -1.52094855e-01  1.62786131e-01 -1.31893088e-01 -7.46596562e-02
 -6.62174174e-02 -1.51763021e-01 -1.29946455e-02 -5.70321256e-02
  2.99272899e-02 -2.60476078e-02 -1.90180181e-02  7.34815014e-02
  1.47633207e-01  4.82952862e-02 -3.97547116e-02  1.51762323e-01
 -1.68855684e-01 -4.63765307e-02  3.66527179e-01 -2.11988025e-01
  5.05282797e-02 -1.62505886e-02  3.90122288e-02  6.42406537e-02
  2.62788841e-01 -2.46255119e-01 -1.77690244e-01 -5.74890468e-02
 -1.05123497e-01 -6.97348497e-02 -2.68585654e-03 -7.21580133e-02
 -5.20239324e-03  8.41146074e-02 -2.15252524e-02 -2.72611966e-02
 -7.19985564e-02 -7.52937238e-02 -5.03551980e-02 -5.58924884e-02]
```