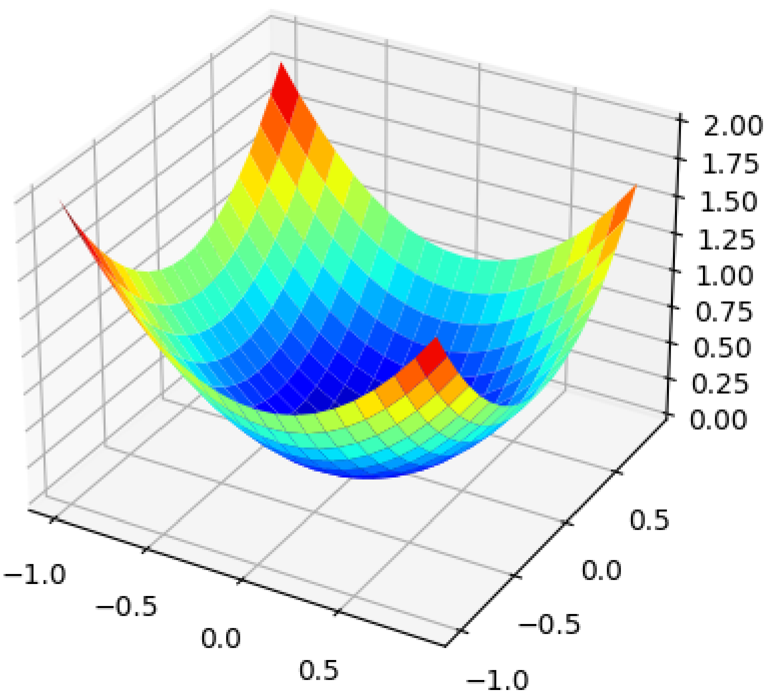
Gaurav Kedia

In [12]:
```
# 3d plot of the test function
from numpy import arange
from numpy import meshgrid
from matplotlib import pyplot
import matplotlib.pyplot as plt
import numpy as np
# objective function
def objective(x, y):
     return x**2.0 + y**2.0

# define range for input
r_min, r_max = -1.0, 1.0
# sample input range uniformly at 0.1 increments
xaxis = arange(r_min, r_max, 0.1)
yaxis = arange(r_min, r_max, 0.1)
# create a mesh from the axis
x, y = meshgrid(xaxis, yaxis)
# compute targets
results = objective(x, y)
# create a surface plot with the jet color scheme
figure = plt.figure()
axis = figure.add_subplot(projection='3d')
axis.plot_surface(x, y, results, cmap='jet')
# show the plot
pyplot.show()
# print(np.shape(xaxis))
print(results)
# print(xaxis)
# print(y)
```



```
[[2.00000000e+00 1.81000000e+00 1.64000000e+00 1.49000000e+00
  1.36000000e+00 1.25000000e+00 1.16000000e+00 1.09000000e+00
  1.04000000e+00 1.01000000e+00 1.00000000e+00 1.01000000e+00
  1.04000000e+00 1.09000000e+00 1.16000000e+00 1.25000000e+00
  1.36000000e+00 1.49000000e+00 1.64000000e+00 1.81000000e+00]
 [1.81000000e+00 1.62000000e+00 1.45000000e+00 1.30000000e+00
  1.17000000e+00 1.06000000e+00 9.70000000e-01 9.00000000e-01
  8.50000000e-01 8.20000000e-01 8.10000000e-01 8.20000000e-01
  8.50000000e-01 9.00000000e-01 9.70000000e-01 1.06000000e+00
  1.17000000e+00 1.30000000e+00 1.45000000e+00 1.62000000e+00]
 [1.64000000e+00 1.45000000e+00 1.28000000e+00 1.13000000e+00
  1.00000000e+00 8.90000000e-01 8.00000000e-01 7.30000000e-01
  6.80000000e-01 6.50000000e-01 6.40000000e-01 6.50000000e-01
  6.80000000e-01 7.30000000e-01 8.00000000e-01 8.90000000e-01
  1.00000000e+00 1.13000000e+00 1.28000000e+00 1.45000000e+00]
 [1.49000000e+00 1.30000000e+00 1.13000000e+00 9.80000000e-01
  8.50000000e-01 7.40000000e-01 6.50000000e-01 5.80000000e-01
  5.30000000e-01 5.00000000e-01 4.90000000e-01 5.00000000e-01
  5.30000000e-01 5.80000000e-01 6.50000000e-01 7.40000000e-01
  8.50000000e-01 9.80000000e-01 1.13000000e+00 1.30000000e+00]
 [1.36000000e+00 1.17000000e+00 1.00000000e+00 8.50000000e-01
  7.20000000e-01 6.10000000e-01 5.20000000e-01 4.50000000e-01
  4.00000000e-01 3.70000000e-01 3.60000000e-01 3.70000000e-01
  4.00000000e-01 4.50000000e-01 5.20000000e-01 6.10000000e-01
  7.20000000e-01 8.50000000e-01 1.00000000e+00 1.17000000e+00]
 [1.25000000e+00 1.06000000e+00 8.90000000e-01 7.40000000e-01
  6.10000000e-01 5.00000000e-01 4.10000000e-01 3.40000000e-01
  2.90000000e-01 2.60000000e-01 2.50000000e-01 2.60000000e-01
  2.90000000e-01 3.40000000e-01 4.10000000e-01 5.00000000e-01
  6.10000000e-01 7.40000000e-01 8.90000000e-01 1.06000000e+00]
 [1.16000000e+00 9.70000000e-01 8.00000000e-01 6.50000000e-01
  5.20000000e-01 4.10000000e-01 3.20000000e-01 2.50000000e-01
  2.00000000e-01 1.70000000e-01 1.60000000e-01 1.70000000e-01
  2.00000000e-01 2.50000000e-01 3.20000000e-01 4.10000000e-01
  5.20000000e-01 6.50000000e-01 8.00000000e-01 9.70000000e-01]
 [1.09000000e+00 9.00000000e-01 7.30000000e-01 5.80000000e-01
  4.50000000e-01 3.40000000e-01 2.50000000e-01 1.80000000e-01
  1.30000000e-01 1.00000000e-01 9.00000000e-02 1.00000000e-01
  1.30000000e-01 1.80000000e-01 2.50000000e-01 3.40000000e-01
  4.50000000e-01 5.80000000e-01 7.30000000e-01 9.00000000e-01]
 [1.04000000e+00 8.50000000e-01 6.80000000e-01 5.30000000e-01
  4.00000000e-01 2.90000000e-01 2.00000000e-01 1.30000000e-01
  8.00000000e-02 5.00000000e-02 4.00000000e-02 5.00000000e-02
  8.00000000e-02 1.30000000e-01 2.00000000e-01 2.90000000e-01
  4.00000000e-01 5.30000000e-01 6.80000000e-01 8.50000000e-01]
 [1.01000000e+00 8.20000000e-01 6.50000000e-01 5.00000000e-01
  3.70000000e-01 2.60000000e-01 1.70000000e-01 1.00000000e-01
  5.00000000e-02 2.00000000e-02 1.00000000e-02 2.00000000e-02
  5.00000000e-02 1.00000000e-01 1.70000000e-01 2.60000000e-01
  3.70000000e-01 5.00000000e-01 6.50000000e-01 8.20000000e-01]
 [1.00000000e+00 8.10000000e-01 6.40000000e-01 4.90000000e-01
  3.60000000e-01 2.50000000e-01 1.60000000e-01 9.00000000e-02
  4.00000000e-02 1.00000000e-02 9.86076132e-32 1.00000000e-02
  4.00000000e-02 9.00000000e-02 1.60000000e-01 2.50000000e-01
  3.60000000e-01 4.90000000e-01 6.40000000e-01 8.10000000e-01]
 [1.01000000e+00 8.20000000e-01 6.50000000e-01 5.00000000e-01
  3.70000000e-01 2.60000000e-01 1.70000000e-01 1.00000000e-01
  5.00000000e-02 2.00000000e-02 1.00000000e-02 2.00000000e-02
  5.00000000e-02 1.00000000e-01 1.70000000e-01 2.60000000e-01
  3.70000000e-01 5.00000000e-01 6.50000000e-01 8.20000000e-01]
```

```
[1.04000000e+00 8.50000000e-01 6.80000000e-01 5.30000000e-01
 4.00000000e-01 2.90000000e-01 2.00000000e-01 1.30000000e-01
 8.00000000e-02 5.00000000e-02 4.00000000e-02 5.00000000e-02
 8.00000000e-02 1.30000000e-01 2.00000000e-01 2.90000000e-01
 4.00000000e-01 5.30000000e-01 6.80000000e-01 8.50000000e-01]
[1.09000000e+00 9.00000000e-01 7.30000000e-01 5.80000000e-01
 4.50000000e-01 3.40000000e-01 2.50000000e-01 1.80000000e-01
 1.30000000e-01 1.00000000e-01 9.00000000e-02 1.00000000e-01
 1.30000000e-01 1.80000000e-01 2.50000000e-01 3.40000000e-01
 4.50000000e-01 5.80000000e-01 7.30000000e-01 9.00000000e-01]
[1.16000000e+00 9.70000000e-01 8.00000000e-01 6.50000000e-01
 5.20000000e-01 4.10000000e-01 3.20000000e-01 2.50000000e-01
 2.00000000e-01 1.70000000e-01 1.60000000e-01 1.70000000e-01
 2.00000000e-01 2.50000000e-01 3.20000000e-01 4.10000000e-01
 5.20000000e-01 6.50000000e-01 8.00000000e-01 9.70000000e-01]
[1.25000000e+00 1.06000000e+00 8.90000000e-01 7.40000000e-01
 6.10000000e-01 5.00000000e-01 4.10000000e-01 3.40000000e-01
 2.90000000e-01 2.60000000e-01 2.50000000e-01 2.60000000e-01
 2.90000000e-01 3.40000000e-01 4.10000000e-01 5.00000000e-01
 6.10000000e-01 7.40000000e-01 8.90000000e-01 1.06000000e+00]
[1.36000000e+00 1.17000000e+00 1.00000000e+00 8.50000000e-01
 7.20000000e-01 6.10000000e-01 5.20000000e-01 4.50000000e-01
 4.00000000e-01 3.70000000e-01 3.60000000e-01 3.70000000e-01
 4.00000000e-01 4.50000000e-01 5.20000000e-01 6.10000000e-01
 7.20000000e-01 8.50000000e-01 1.00000000e+00 1.17000000e+00]
[1.49000000e+00 1.30000000e+00 1.13000000e+00 9.80000000e-01
 8.50000000e-01 7.40000000e-01 6.50000000e-01 5.80000000e-01
 5.30000000e-01 5.00000000e-01 4.90000000e-01 5.00000000e-01
 5.30000000e-01 5.80000000e-01 6.50000000e-01 7.40000000e-01
 8.50000000e-01 9.80000000e-01 1.13000000e+00 1.30000000e+00]
[1.64000000e+00 1.45000000e+00 1.28000000e+00 1.13000000e+00
 1.00000000e+00 8.90000000e-01 8.00000000e-01 7.30000000e-01
 6.80000000e-01 6.50000000e-01 6.40000000e-01 6.50000000e-01
 6.80000000e-01 7.30000000e-01 8.00000000e-01 8.90000000e-01
 1.00000000e+00 1.13000000e+00 1.28000000e+00 1.45000000e+00]
[1.81000000e+00 1.62000000e+00 1.45000000e+00 1.30000000e+00
 1.17000000e+00 1.06000000e+00 9.70000000e-01 9.00000000e-01
 8.50000000e-01 8.20000000e-01 8.10000000e-01 8.20000000e-01
 8.50000000e-01 9.00000000e-01 9.70000000e-01 1.06000000e+00
 1.17000000e+00 1.30000000e+00 1.45000000e+00 1.62000000e+00]]
```

In [6]:

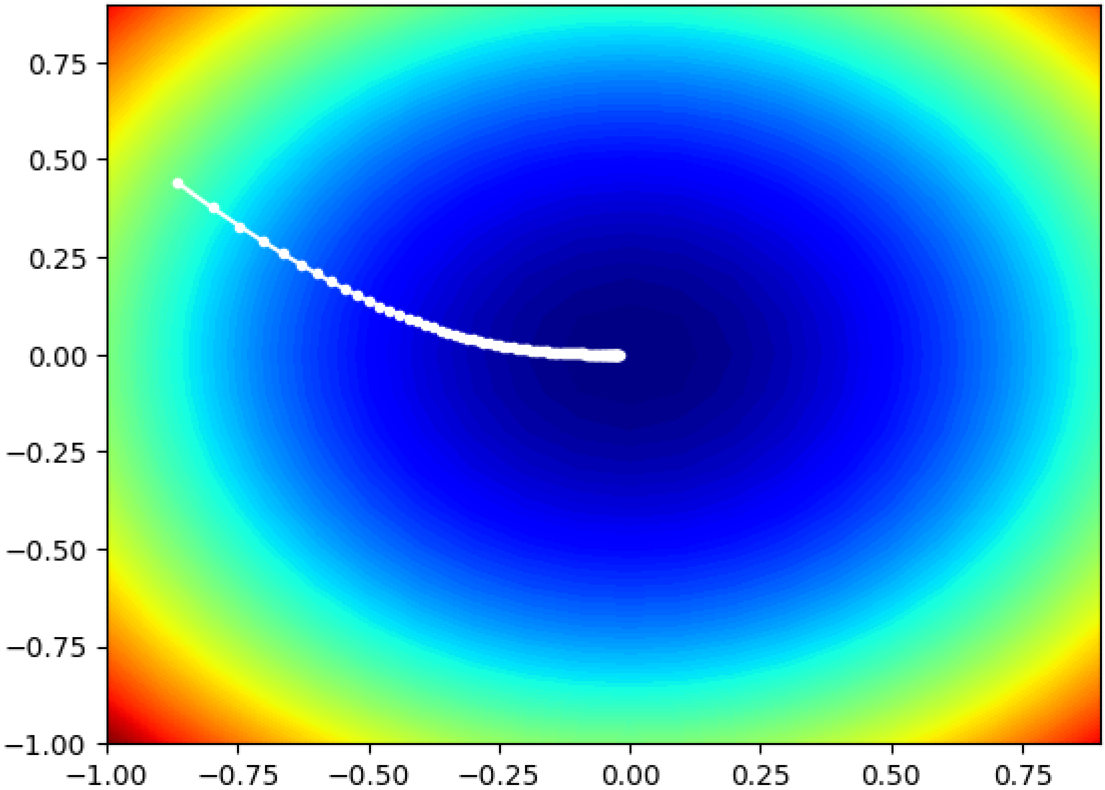
```python
# gradient descent optimization with adagrad for a two-dimensional test function
from math import sqrt
from numpy import asarray
from numpy.random import rand
from numpy.random import seed
from numpy import arange
from numpy import meshgrid
from matplotlib import pyplot
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

# derivative of objective function
def derivative(x, y):
    return asarray([x * 2.0, y * 2.0])


# gradient descent algorithm with adagrad
def adagrad(objective, derivative, bounds, n_iter, step_size):
    solutions = list()
```

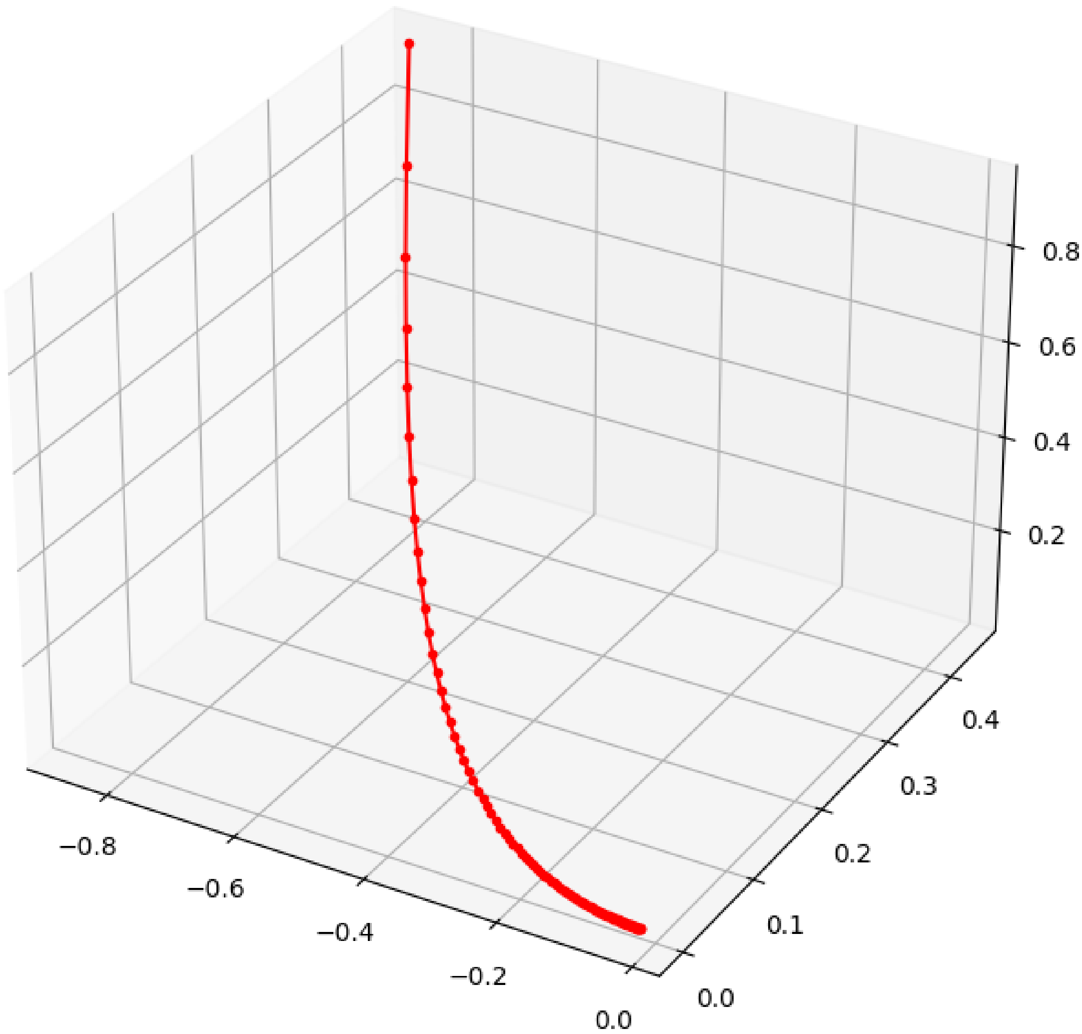
```
    score=list()
    solution = bounds[:, 0] + rand(len(bounds)) * (bounds[:, 1] - bounds[:, 0])
    solution=np.array( [-0.96595599, 0.54064899])
    print(solution)
    sq_grad_sums = [0.0 for _ in range(bounds.shape[0])]
    for it in range(n_iter):
        gradient = derivative(solution[0], solution[1])
        for i in range(gradient.shape[0]):
            sq_grad_sums[i] += gradient[i]**2.0
        new_solution = list()
        for i in range(solution.shape[0]):
            alpha = step_size / (1e-8 + sqrt(sq_grad_sums[i]))
            value = solution[i] - alpha * gradient[i]
            new_solution.append(value)
        solution = asarray(new_solution)
        solutions.append(solution)
        solution_eval = objective(solution[0], solution[1])
        print('>%d f(%s) = %.5f' % (it, solution, solution_eval))
        score.append(solution_eval)
    return [solutions, score]

seed(1)
bounds = asarray([[-1.0, 1.0], [-1.0, 1.0]])
n_iter = 100
step_size = 0.1
solution, score = adagrad(objective, derivative, bounds, n_iter, step_size)
# print('Done!')
# print('f(%s) = f(%s)' % (solution, score))

xaxis = arange(r_min, r_max, 0.1)
yaxis = arange(r_min, r_max, 0.1)



x, y = meshgrid(xaxis, yaxis)
results = objective(x, y)


pyplot.contourf(x, y, results, levels=100, cmap='jet')
solutions = asarray(solution)
# print(solutions[:, 0])
# print(solutions[:, 1])
pyplot.plot(solutions[:, 0], solutions[:, 1], '.-', color='w')
pyplot.show()



# print(score)
figure =  pyplot.figure(figsize=(10, 8))
axis = figure.add_subplot(projection='3d')
# axis.plot_surface(x, y, results, cmap='jet')
solutions = asarray(solutions)
pyplot.plot(solutions[:, 0], solutions[:, 1],score, '.-', color='r')
pyplot.show()



figure =  pyplot.figure(figsize=(10, 8))
axis = figure.add_subplot(projection='3d')
axis.plot_surface(x, y, results, cmap='jet', alpha=0.8)
```

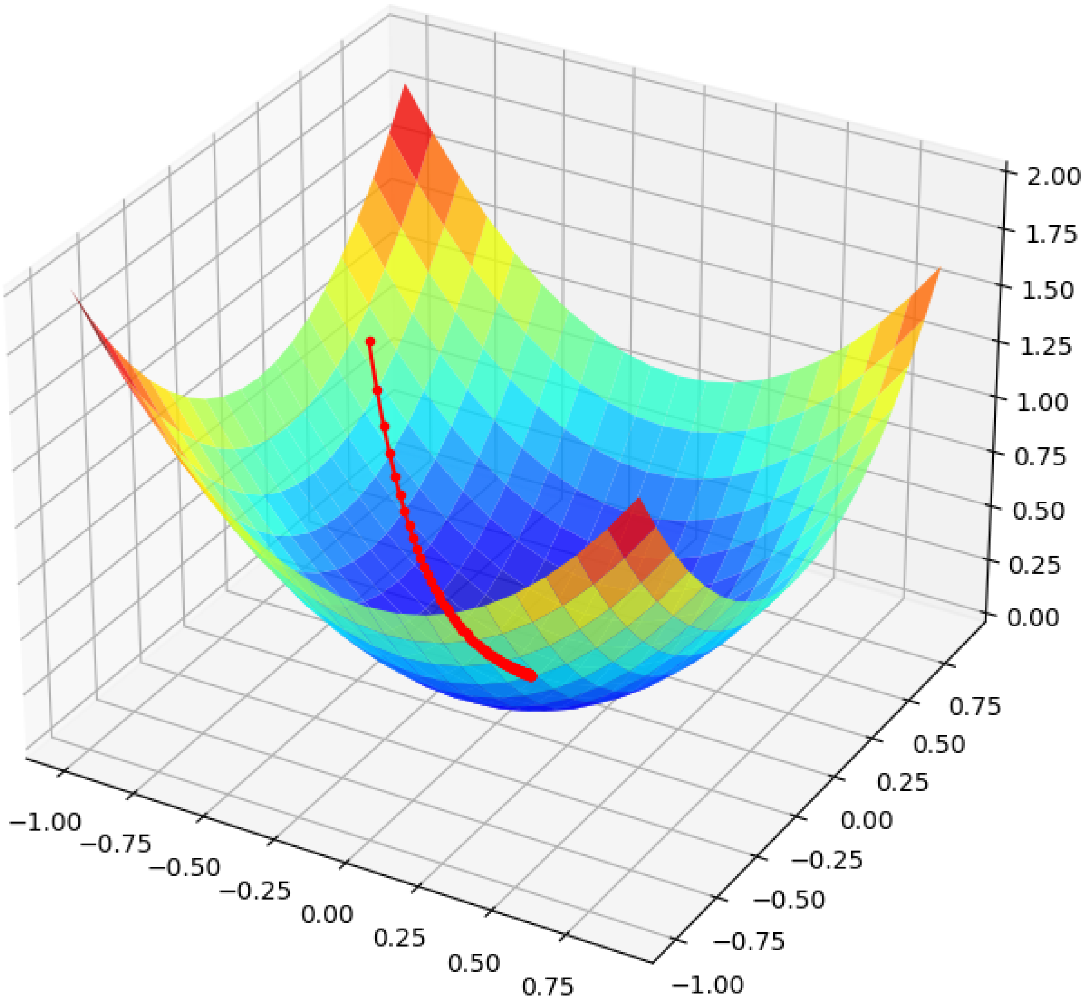
```
solutions = asarray(solutions)
pyplot.plot(solutions[:, 0], solutions[:, 1],score, '.-', color='r', zorder=10)
pyplot.show()
```

```
[-0.96595599  0.54064899]
>0 f([-0.86595599  0.44064899]) = 0.94405
>1 f([-0.79920463  0.37747134]) = 0.78121
>2 f([-0.74675325  0.32987503]) = 0.66646
>3 f([-0.70274519  0.29147005]) = 0.57881
>4 f([-0.66448221  0.25933601]) = 0.50879
>5 f([-0.63046077  0.23184622]) = 0.45123
>6 f([-0.59974197  0.20798051]) = 0.40295
>7 f([-0.57169298  0.18704587]) = 0.36182
>8 f([-0.54586313  0.16854351]) = 0.32637
>9 f([-0.5219178   0.15209838]) = 0.29553
>10 f([-0.49960032  0.1374186 ]) = 0.26848
>11 f([-0.47870856  0.12427078]) = 0.24461
>12 f([-0.45907986  0.11246407]) = 0.22340
>13 f([-0.44058089  0.10183956]) = 0.20448
>14 f([-0.42310069  0.09226298]) = 0.18753
>15 f([-0.40654566  0.08361941]) = 0.17227
>16 f([-0.39083591  0.07580953]) = 0.15850
>17 f([-0.37590256  0.06874675]) = 0.14603
>18 f([-0.36168569  0.06235508]) = 0.13470
>19 f([-0.34813273  0.05656739]) = 0.12440
>20 f([-0.33519721  0.05132412]) = 0.11499
>21 f([-0.32283784  0.04657222]) = 0.10639
>22 f([-0.31101762  0.04226429]) = 0.09852
>23 f([-0.29970331  0.03835783]) = 0.09129
>24 f([-0.28886482  0.03481466]) = 0.08465
>25 f([-0.27847484  0.03160045]) = 0.07855
>26 f([-0.26850844  0.02868422]) = 0.07292
>27 f([-0.25894279  0.02603804]) = 0.06773
>28 f([-0.24975692  0.02363667]) = 0.06294
>29 f([-0.24093149  0.02145729]) = 0.05851
>30 f([-0.2324486   0.01947924]) = 0.05441
>31 f([-0.22429165  0.01768382]) = 0.05062
>32 f([-0.21644521  0.01605411]) = 0.04711
>33 f([-0.20889488  0.01457475]) = 0.04385
>34 f([-0.20162719  0.01323183]) = 0.04083
>35 f([-0.19462956  0.01201274]) = 0.03802
>36 f([-0.18789014  0.01090604]) = 0.03542
>37 f([-0.18139781  0.00990134]) = 0.03300
>38 f([-0.17514209  0.00898924]) = 0.03076
>39 f([-0.1691131   0.00816119]) = 0.02867
>40 f([-0.16330148  0.00740943]) = 0.02672
>41 f([-0.1576984   0.00672694]) = 0.02491
>42 f([-0.15229547  0.00610733]) = 0.02323
>43 f([-0.14708474  0.0055448 ]) = 0.02166
>44 f([-0.14205866  0.00503408]) = 0.02021
>45 f([-0.13721003  0.00457041]) = 0.01885
>46 f([-0.13253201  0.00414946]) = 0.01758
>47 f([-0.1280181   0.00376727]) = 0.01640
>48 f([-0.12366206  0.00342029]) = 0.01530
>49 f([-0.11945796  0.00310527]) = 0.01428
>50 f([-0.11540014  0.00281927]) = 0.01333
>51 f([-0.11148316  0.00255961]) = 0.01244
>52 f([-0.10770184  0.00232386]) = 0.01161
>53 f([-0.10405121  0.00210983]) = 0.01083
>54 f([-0.10052651  0.00191551]) = 0.01011
>55 f([-0.09712319  0.00173909]) = 0.00944
>56 f([-0.09383686  0.00157892]) = 0.00881
>57 f([-0.09066333  0.0014335 ]) = 0.00822
>58 f([-0.08759856  0.00130147]) = 0.00768
```

```
>59 f([-0.08463869  0.0011816 ]) = 0.00717
>60 f([-0.08178001  0.00107277]) = 0.00669
>61 f([-0.07901893  0.00097397]) = 0.00624
>62 f([-0.07635202  0.00088427]) = 0.00583
>63 f([-0.07377597  0.00080282]) = 0.00544
>64 f([-0.07128761  0.00072888]) = 0.00508
>65 f([-0.06888387  0.00066175]) = 0.00475
>66 f([-0.0665618  0.0006008]) = 0.00443
>67 f([-0.06431858  0.00054547]) = 0.00414
>68 f([-0.06215147  0.00049523]) = 0.00386
>69 f([-0.06005784  0.00044962]) = 0.00361
>70 f([-0.05803514  0.00040821]) = 0.00337
>71 f([-0.05608094  0.00037061]) = 0.00315
>72 f([-0.05419288  0.00033648]) = 0.00294
>73 f([-0.05236869  0.00030549]) = 0.00274
>74 f([-0.05060618  0.00027735]) = 0.00256
>75 f([-0.04890323  0.00025181]) = 0.00239
>76 f([-0.04725781  0.00022862]) = 0.00223
>77 f([-0.04566795  0.00020756]) = 0.00209
>78 f([-0.04413177  0.00018844]) = 0.00195
>79 f([-0.04264742  0.00017109]) = 0.00182
>80 f([-0.04121314  0.00015533]) = 0.00170
>81 f([-0.03982723  0.00014102]) = 0.00159
>82 f([-0.03848805  0.00012804]) = 0.00148
>83 f([-0.037194    0.00011624]) = 0.00138
>84 f([-0.03594357  0.00010554]) = 0.00129
>85 f([-3.47352546e-02  9.58175969e-05]) = 0.00121
>86 f([-3.35676427e-02  8.69926931e-05]) = 0.00113
>87 f([-3.24393515e-02  7.89805725e-05]) = 0.00105
>88 f([-3.13490497e-02  7.17063768e-05]) = 0.00098
>89 f([-3.02954520e-02  6.51021424e-05]) = 0.00092
>90 f([-2.92773170e-02  5.91061651e-05]) = 0.00086
>91 f([-2.82934459e-02  5.36624238e-05]) = 0.00080
>92 f([-2.73426810e-02  4.87200569e-05]) = 0.00075
>93 f([-2.64239042e-02  4.42328872e-05]) = 0.00070
>94 f([-2.55360353e-02  4.01589908e-05]) = 0.00065
>95 f([-2.46780313e-02  3.64603046e-05]) = 0.00061
>96 f([-2.38488845e-02  3.31022713e-05]) = 0.00057
>97 f([-2.30476216e-02  3.00535166e-05]) = 0.00053
>98 f([-2.22733023e-02  2.72855554e-05]) = 0.00050
>99 f([-2.15250184e-02  2.47725264e-05]) = 0.00046
```


0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75


0.8
0.6
0.4
0.2
0.4
0.3
0.2
0.1
0.0
−0.8
−0.6
−0.4
−0.2
0.0


2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

In [ ]: