In [ ]:
```
from utils import *
```

Nelder-Mead method

12. Implement the two and three dimensional versions of Nelder-Mead method. Use a programming language of your choice.

One step beyond: Why fix the dimension? Surely you can pass the dimension as a parameter or better still compute it on the fly.

13. Qualitatively compare Nelder-Mead method with the methods you have implemented in HW2/3,4 (GD, Polyak GD, Nesterov GD, AdaGrad GD, Newton, BFGS) on

a)

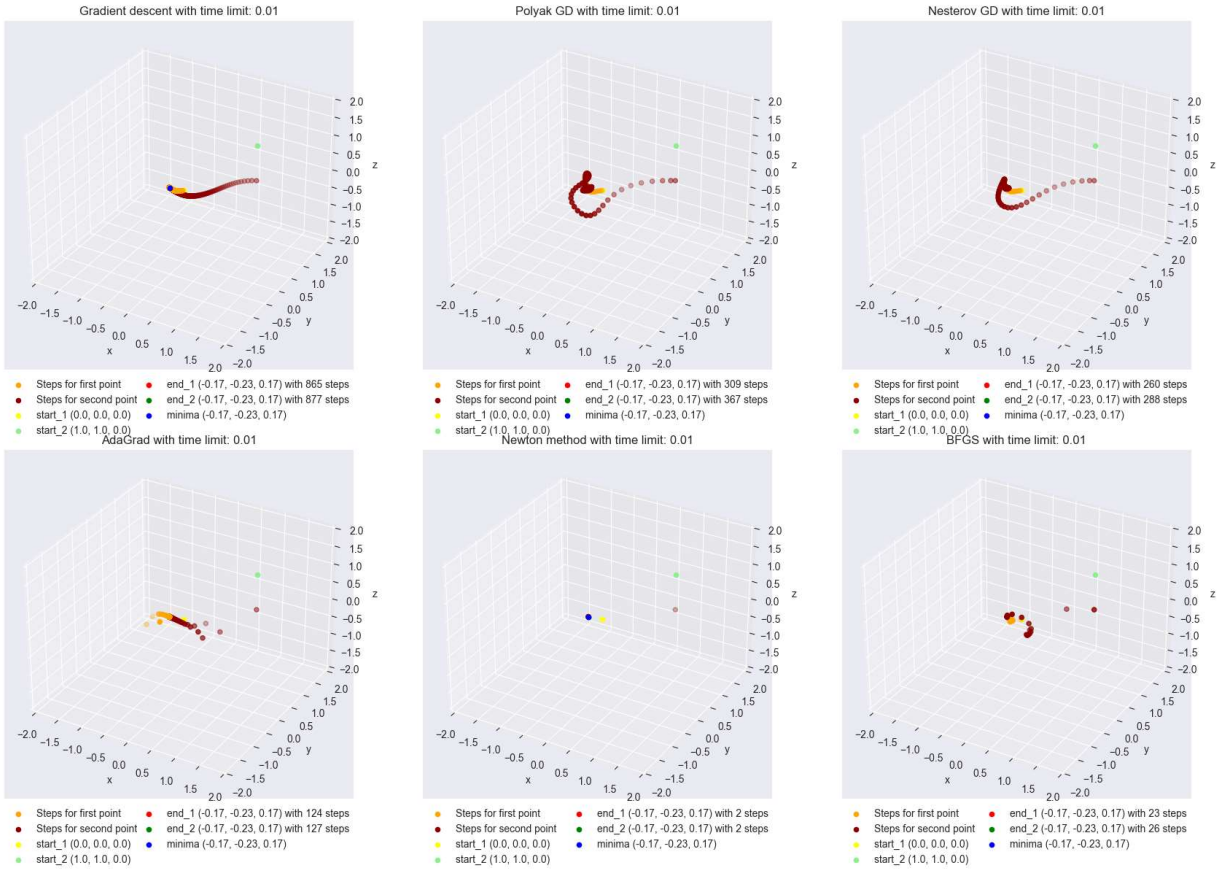
$$f(x,y,z) = (x-z)^2 + (2y+z)^2 + (4x-2y+z)^2 + x + y$$

With initial points (0, 0, 0) and (1, 1, 0) and actual minima at f(-1/6, -11/48, 1/6) = 19/96

In [ ]:
```
x0_1 = np.array([0,0,0])
x0_2 = np.array([1,1,0])
x_min = np.array([-1/6,-11/48,1/6])
gammas = np.array([[0.005, 0.005, 0.005, 0.5],
                   [0.005, 0.005, 0.005, 0.5]])
v = 0.9

optimizing_functions = ['Gradient descent', 'Polyak GD', 'Nesterov GD', 'AdaGrad', '

diameters = [0.5, 1, 2]
```

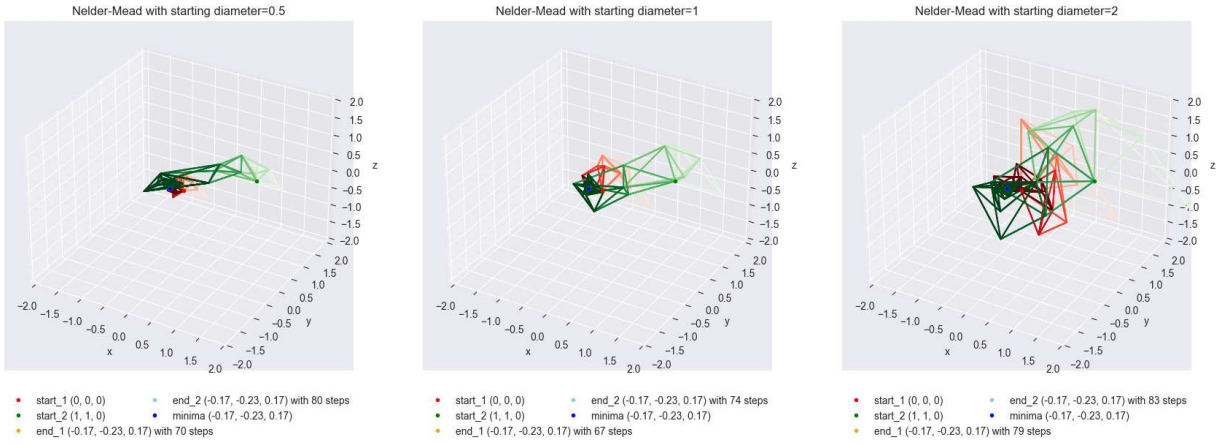
In [ ]:
```
plot_all(x0_1, x0_2, x_min, gammas, v, 1000, 0.01, optimizing_functions, f1, 1)
```

```
Difference between final function value and minima function value:

Gradient descent for starting point (0.0, 0.0, 0.0): 1.5739354264354688e-12
Gradient descent for starting point (1.0, 1.0): 1.6802031987950272e-11
Polyak GD for starting point (0.0, 0.0, 0.0): 1.27675647831893e-15
Polyak GD for starting point (1.0, 1.0): 1.1102230246251565e-16
Nesterov GD for starting point (0.0, 0.0, 0.0): 3.6637359812630166e-15
Nesterov GD for starting point (1.0, 1.0): 1.7208456881689926e-15
AdaGrad for starting point (0.0, 0.0, 0.0): 5.551115123125783e-17
AdaGrad for starting point (1.0, 1.0): 8.326672684688674e-17
Newton method for starting point (0.0, 0.0, 0.0): 5.551115123125783e-17
Newton method for starting point (1.0, 1.0): 0.0
BFGS for starting point (0.0, 0.0, 0.0): 2.7755575615628914e-17
BFGS for starting point (1.0, 1.0): 0.0
```



In [ ]:

```
plot_nelder_mead_3D(x0_1, x0_2, x_min, f1, diameters, time_limit=0.01)
```



For this function all descent methods and Nelder-Mean method converge in 0.01 seconds for both initial points. We can see that Nelder-Mead converges in approximatelly 110 steps for all diameters, which is lower than any descent method. The Newton method and BFGS still perform the best for this function.

b)

$$f(x,y,z) = (x-1)^2 + (y-1)^2 + 100\left(y-x^2\right)^2 + 100\left(z-y^2\right)^2$$

With initial points (1.2, 1.2, 1.2) and (−1, 1.2, 1.2) and actual minima at f(1, 1, 1) = 0

In [ ]:

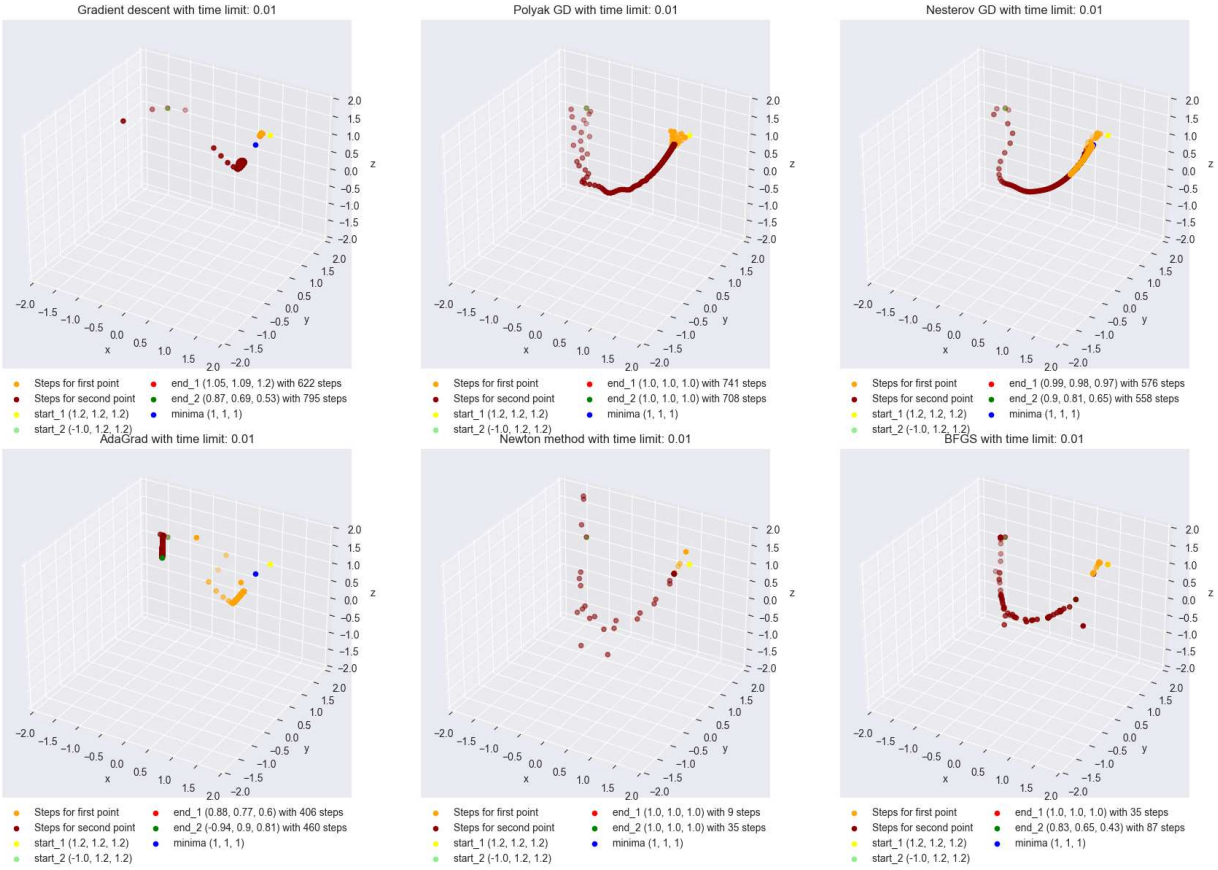
```
x0_1 = np.array([1.2, 1.2, 1.2])
x0_2 = np.array([-1, 1.2, 1.2])
x_min = np.array([1,1,1])

gammas = np.array([[0.001, 0.001, 0.001, 1],
```

```
                    [0.002, 0.002, 0.0011, 0.1]])
v = 0.9

optimizing_functions = ['Gradient descent', 'Polyak GD', 'Nesterov GD', 'AdaGrad', '
diameters = [0.5, 1, 2]
```

In [ ]:

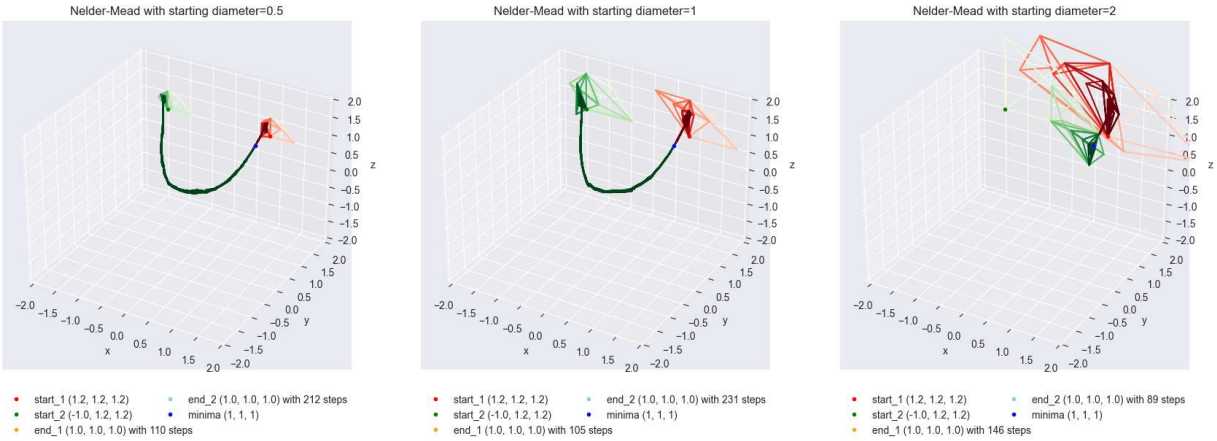
```
plot_all(x0_1, x0_2, x_min, gammas, v, 1000, 0.01, optimizing_functions, f2, 2)
```

```
Difference between final function value and minima function value:

Gradient descent for starting point (1.2, 1.2, 1.2): 0.011108825196933547
Gradient descent for starting point (-1.0, 1.2): 0.7881068471631951
Polyak GD for starting point (1.2, 1.2, 1.2): 1.1863776050603093e-06
Polyak GD for starting point (-1.0, 1.2): 4.5217890107326075e-08
Nesterov GD for starting point (1.2, 1.2, 1.2): 0.0003517231237504022
Nesterov GD for starting point (-1.0, 1.2): 0.04629358160587216
AdaGrad for starting point (1.2, 1.2, 1.2): 0.0657624356671844
AdaGrad for starting point (-1.0, 1.2): 3.796988728909847
Newton method for starting point (1.2, 1.2, 1.2): 0.0
Newton method for starting point (-1.0, 1.2): 0.0
BFGS for starting point (1.2, 1.2, 1.2): 8.260969493971449e-24
BFGS for starting point (-1.0, 1.2): 0.3123065096890436
```



In [ ]:

```
plot_nelder_mead_3D(x0_1, x0_2, x_min, f2, diameters, time_limit=0.01)
```



Nelder-Mead method once again converges for all 3 diameters and initial points in less than 0.01 seconds. It outperforms descent methods in terms of steps (normal GD and Adagrad do not even converge in this time limit). The newton method and BFGS still outperform Nelder-Mead in terms of number of steps.

c)

$$f(x,y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2$$

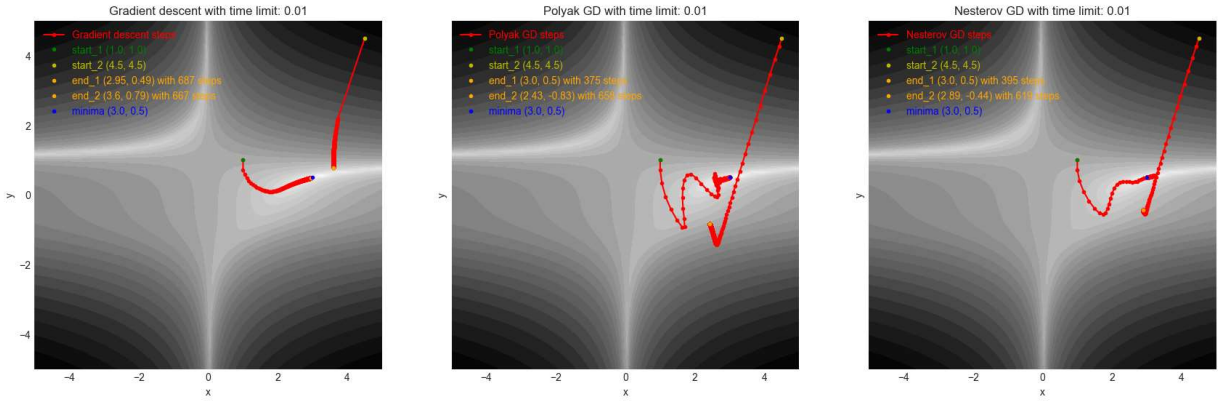
With initial points (1, 1) and (4.5, 4.5) and actual minima at (3, 0.5)

NOTE: To better show contours we plot the function in logspace (minima stays at the same position)

In [ ]:
```
x0_1 = np.array([1, 1])
x0_2 = np.array([4.5, 4.5])
x_min = np.array([3, 0.5])
gammas = np.array([[0.01, 0.01, 0.01, 1],
                   [0.00001, 0.000001, 0.000001, 1]])
v = 0.9
optimizing_functions = ['Gradient descent', 'Polyak GD', 'Nesterov GD', 'AdaGrad', '
diameters = [0.5, 1, 2]
```

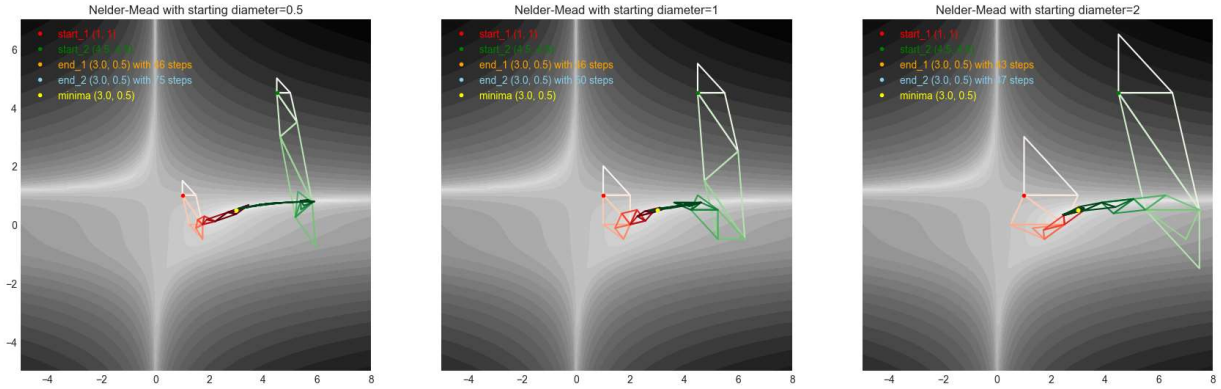
In [ ]:
```
plot_all(x0_1, x0_2, x_min, gammas, v, 1000, 0.01, optimizing_functions, f3, 3)
```

```
Difference between final function value and minima function value:

Gradient descent for starting point (1.0, 1.0): 0.0003863961736320351
Gradient descent for starting point (4.5, 4.5): 1.9399899629671133
Polyak GD for starting point (1.0, 1.0): 1.7213991934635063e-16
Polyak GD for starting point (4.5, 4.5): 12.206952723003166
Nesterov GD for starting point (1.0, 1.0): 3.695762186844486e-17
Nesterov GD for starting point (4.5, 4.5): 7.335777202603242
AdaGrad for starting point (1.0, 1.0): 1.1283271779677551e-13
AdaGrad for starting point (4.5, 4.5): 70.66530612924495
Newton method for starting point (1.0, 1.0): 14.203125
Newton method for starting point (4.5, 4.5): 14.203125
BFGS for starting point (1.0, 1.0): 1.2478932111420876e-26
BFGS for starting point (4.5, 4.5): 1.2726545072510855e-30
```



In [ ]:

```
plot_nelder_mead(x0_1, x0_2, x_min, f3, diameters, time_limit=0.01)
```



We can see that Nelder-Mead method converges in less than 100 steps which take less than 0.01 seconds for both initial points and all 3 specified starting diameters. Meanwhile a lot of descent methods still haven't reached the optimal minima in 0.01 seconds.