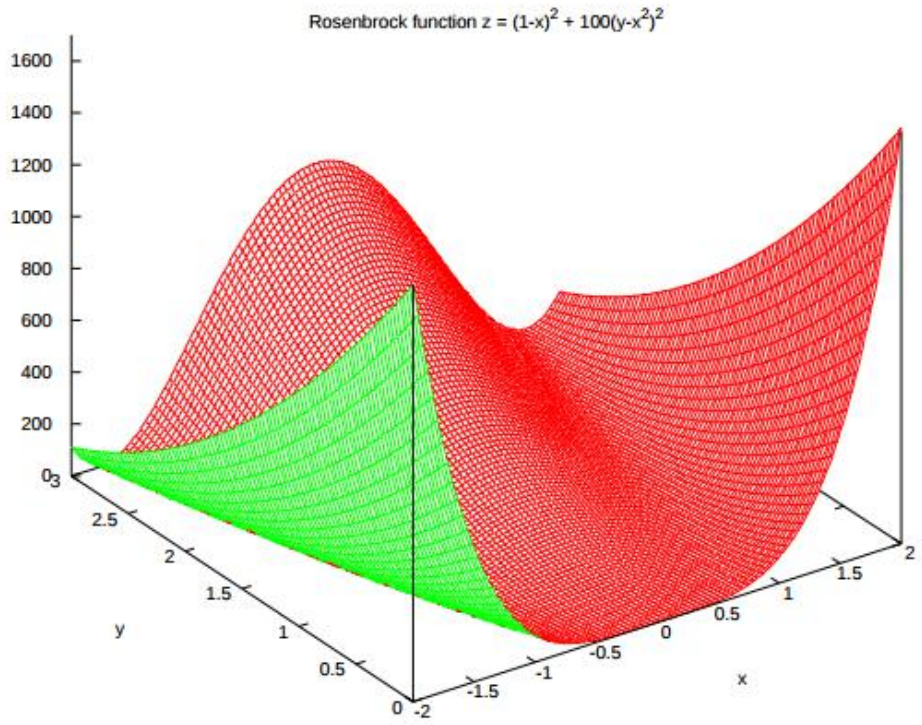
21 O. Znajdź numerycznie (analitycznie zrobić można to bardzo łatwo) minimum funkcji Rosenbrocka (zobacz rysunek)



$$f(x,y) = (1-x)^2 + 100(y-x^2)^2 \,. \tag{16}$$

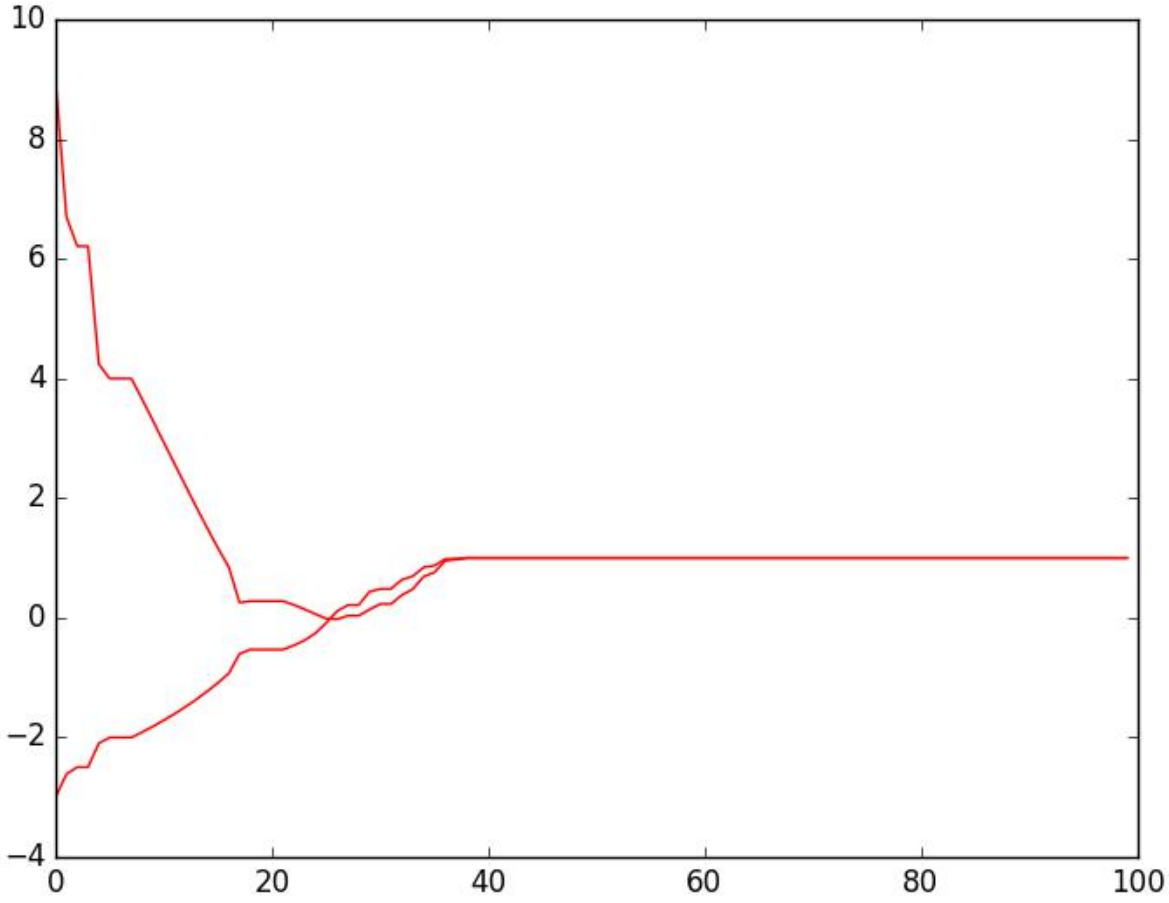
Rozpocznij poszukiwania od kilku–kilkunastu różnych, losowo wybranych punktów i oszacuj, ile trzeba kroków aby zbliżyć się do minimum narozsądną odległość. Przedstaw graficznie drogę, jaką przebywa algorytm poszukujący minimum (to znaczy pokaż położenia kolejnych minimalizacji kierunkowych lub kolejnych zaakceptowanych kroków wykonywanych w metodzie Levenberga–Marquardta).

Zadanie rozwiązane za pomocą algorytmu Levenberga-Marquardta. Funkcje obliczające gradient oraz hessian zostały zaimplementowane przy użyciu funkcji bibliotecznych z klasy numdifftools. Program rozpoczyna działanie z 5 różnych losowych punktów. W każdym z przypadków funkcja osiąga minimum w **punkcie 1.0,1.0, min=0.0.**

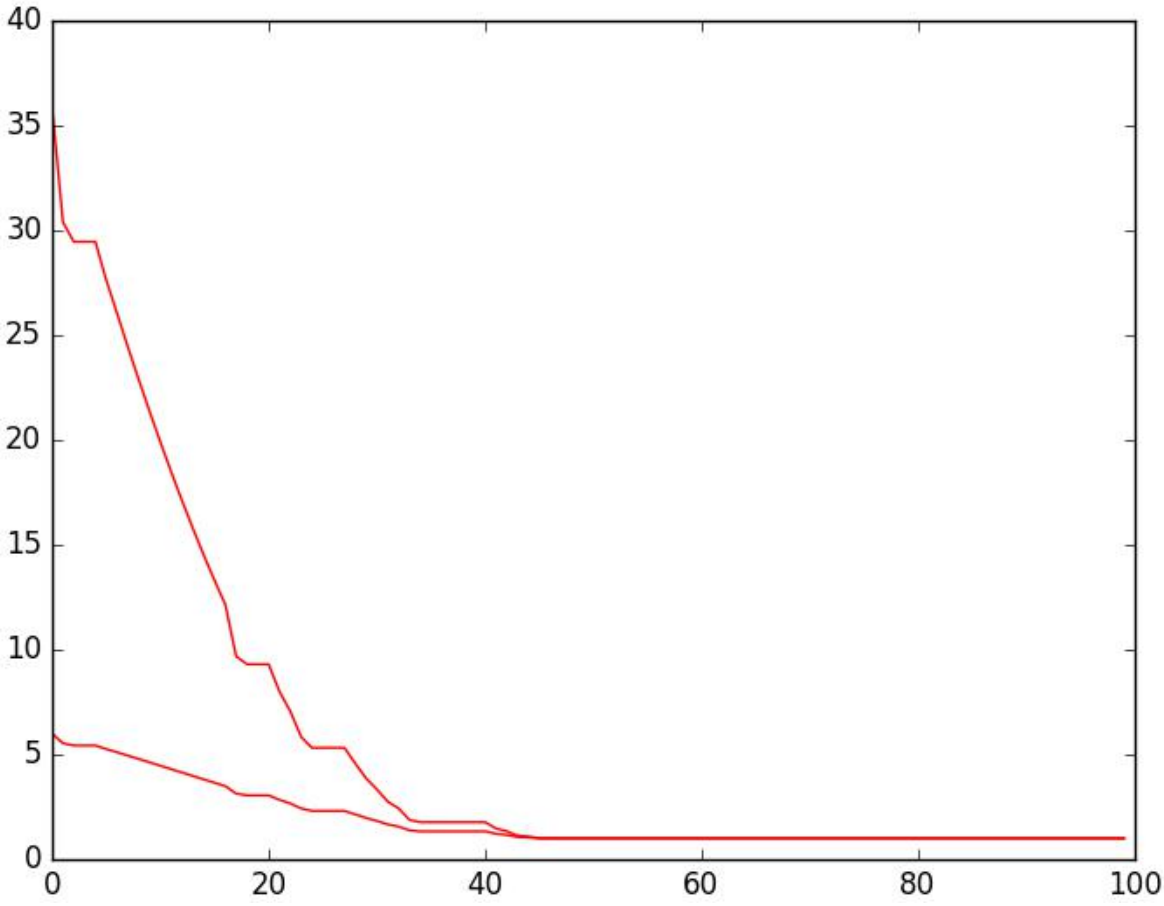
```
import numpy as np
import numdifftools as nd
from numpy.linalg import inv
import matplotlib.pyplot as plt

def LeMa(x):
    f=lambda  x:100*(x[1]- x[0]**2)**2 + (1 - x[0])**2
    l=1/1024
    def Grad(x):
        g = nd.Gradient(f)
        h=g(x)
        return h
```

```
    def Hes(x):
        h=nd.Hessian(f)
        h=h(x)
        h[0, 0] = (1 + l) * h[0, 0]
        h[1, 1] = (1 + l) * h[1, 1]
        return h


    x_test = x - (inv(Hes(x)).dot(Grad(x)))
    iter=100
    solutions=np.zeros((iter,2))
    for i in range(iter):
        grad=Grad(x)

        if(f(x_test)>f(x)):
            l=l*10
            x_test = x - (inv(Hes(x)).dot(grad))

        if (f(x_test) < f(x)):
            l=l/10
            x=x_test.copy()
            x_test = x - (inv(Hes(x)).dot(Grad(x)))
        solutions[i]=x

    args=np.arange(0,iter,1)
    plt.plot(args,solutions,'r')
    plt.show()

point1=np.array([-3,-4])
point2=np.array([6,0])
point3=np.array([-6,4])
point4=np.array([6,6])
point5=np.array([-10,8])

LeMa(point1)
LeMa(point2)
LeMa(point3)
LeMa(point4)
LeMa(point5)
```

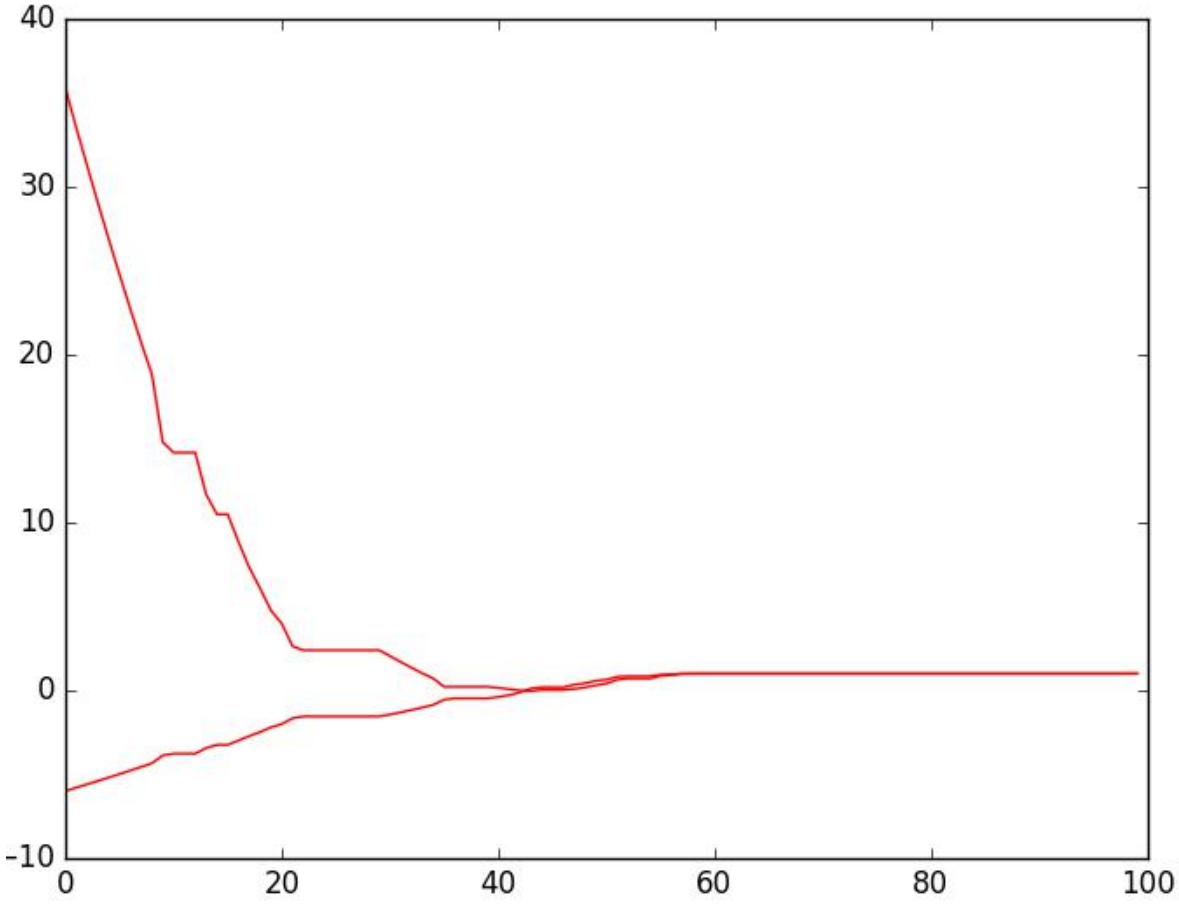
X0=[-3,-4], liczba kroków=ok 38



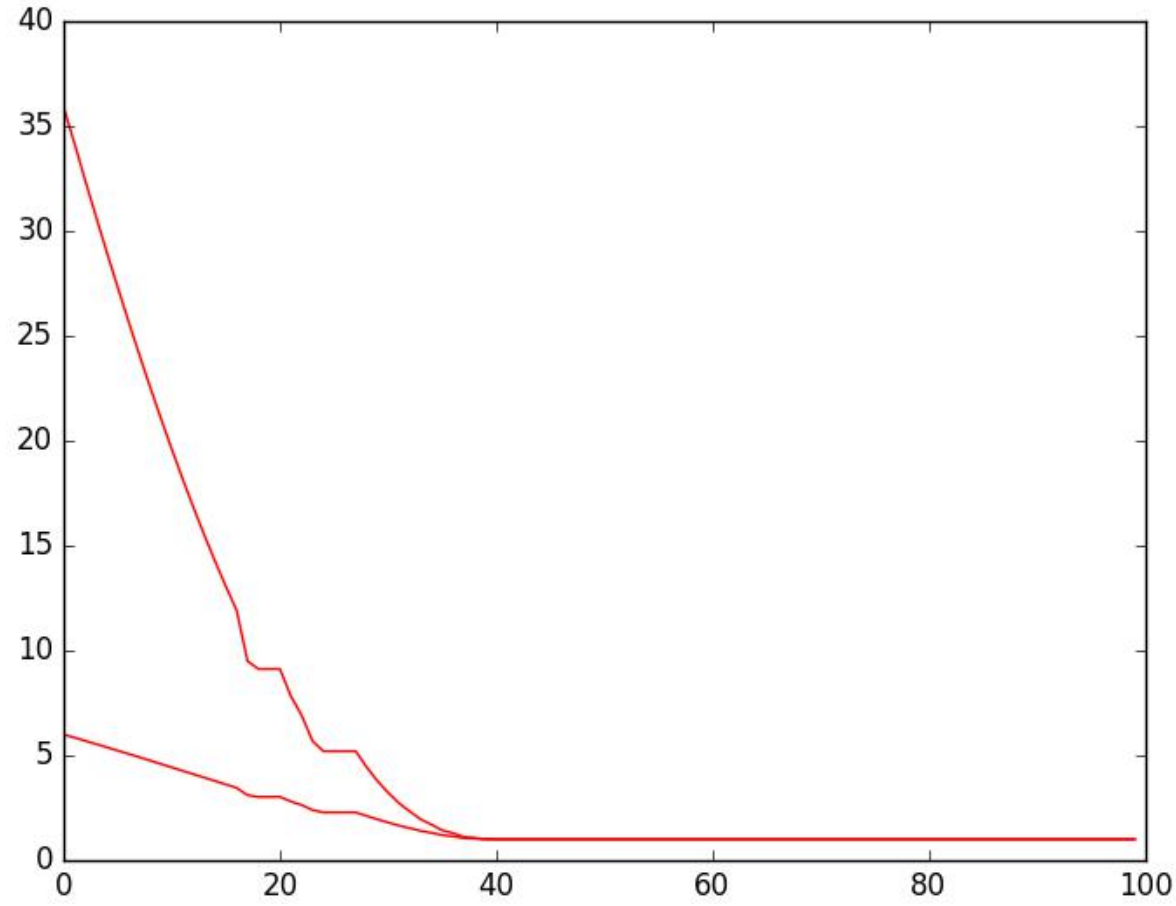
X0=[6,0], liczba kroków=ok 43



X0=[-6,4], liczba kroków=ok 56



X0=[6,6], liczba kroków=ok 38



X0=[-10,8], liczba kroków=ok 80