Курсовая работа по теме

“Градиентный метод минимизации функций многих переменных”

Описание метода

Рассмотрим задачу J(u) -> inf

В основе градиентного метода минимизации функции лежит свойство градиента, а именно: направление наибыстрейшего возрастания функции J(u) в точке u совпадает с направлением градиента J’(u), а направление наибыстрейшего убывания - с направлением антиградиента (-J’(u))

Выберем некоторую начальную точку u0. Градиентный метод заключается в построении последовательности {uk} по правилу:

uk+1 = uk - ak\*J’(uk), ak > 0, k = 0,1... (1)

Где ak - шаг градиентного метода

Если J’(uk) = 0 то uk - стационарная точка. В этом случае процесс (1) прекращается и при необходимости проводится дополнительное исследование поведения функции в окрестности точки uk для выяснения того, достигается ли в uk минимум функции J(u) или нет. Если J(u) - выпуклая функция, то в стационарной точке всегда достигается минимум.

Тесты

Тест 1

F = sin(x) + sin(y)

Начальная точка: x=1, y=1

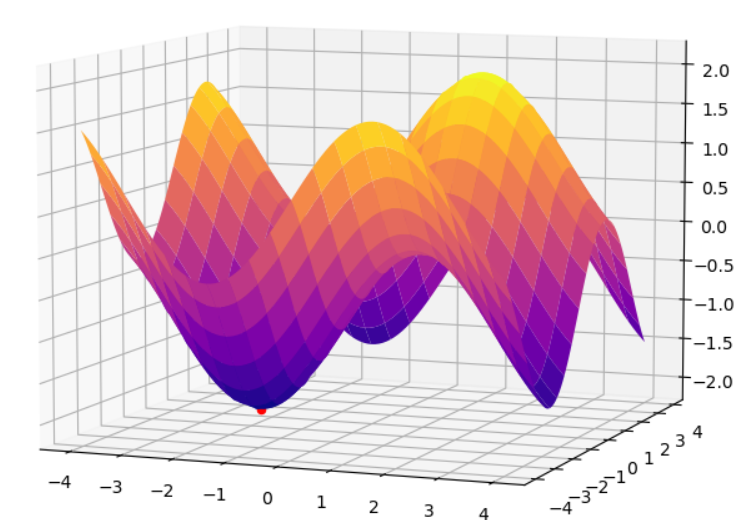
Найденная точка: x=-1.5659426, y=-1.5659426

Значение в этой точке: -1.999976

Достигнутая точность: 9.714e-07

Число итераций: 360

Шаг: 0.01



Тест 2

F = x2 + y2

Начальная точка: x=10, y=10

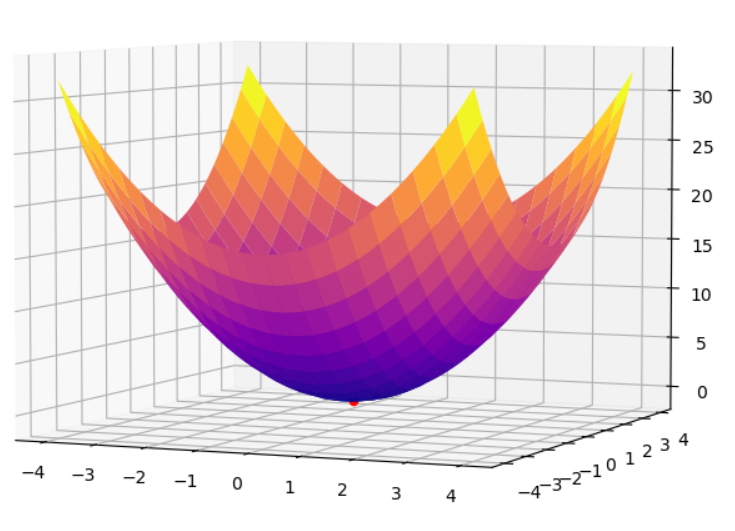
Найденная точка: x=0.002416, y=0.002416

Значение в этой точке: 1.168189e-05

Достигнутая точность: 9.939e-07

Число итераций: 204

Шаг: 0.01



Код программы

import numpy as np

import matplotlib.pyplot as plot

from matplotlib import cm

# градиентный метод минимизации функции многих переменных

a = 0.01 # 0.01

global\_epsilon = 0.000001

chislo = 0 # число итераций

global X

X = np.array((1.0,1.0)) # TODO: подумать как разместить X. Пока его приходится менять для каждого теста

# X - значения иксов. Изначально соотвествуют начальной точке, потом изменяется на каждом шаге

# A - коэффициенты при x.

def function3(X,Y=X[1],T=0):

if T == 0:

F = np.sin(X[0]) + np.sin(X[1]) # 1 тест X = np.array((1.0,1.0))

#F = X[0]\*\*2 + X[1]\*\*2 # 2 тест X = np.array((10.0,10.0))

#F = 0.26 \* (X[0]\*\*2 + X[1]\*\*2) - 0.48 \* X[0] \* X[1] # 3 тест(может стоит заменить) X = np.array((10.0,10.0))

#F = -20.0 \* np.exp(-0.2 \* np.sqrt(0.5 \* (X[0]\*\*2 + X[1]\*\*2))) - np.exp(0.5 \* (np.cos(2 \* np.pi \* X[0]) + np.cos(2 \* np.pi \* X[1]))) + np.e + 20

if T == -1:

F = np.sin(X) + np.sin(Y) # 1 тест X = np.array((1.0,1.0))

#F = X\*\*2 + Y\*\*2 # 2 тест X = np.array((10.0,10.0))

#F = 0.26 \* (X\*\*2 + Y\*\*2) - 0.48 \* X \* Y # 3 тест(может стоит заменить) X = np.array((10.0,10.0))

#F = -20.0 \* np.exp(-0.2 \* np.sqrt(0.5 \* (X\*\*2 + Y\*\*2))) - np.exp(0.5 \* (np.cos(2 \* np.pi \* X) + np.cos(2 \* np.pi \* Y))) + np.e + 20

return F

# eplsilon - значение дифференцируемой переменной

# index - индекс переменной. Остальные значения переменных берутся из массива X

def derivative(epsilon,index):

#return (function1(X,index,epsilon + global\_epsilon) - function1(X,index,epsilon)) / global\_epsilon

return (function3(index,epsilon + global\_epsilon,-1) - function3(index,epsilon,-1)) / global\_epsilon

# TODO: если точность не уменьшается со временем, то вывести сообщение что метод не сходится

# или уменьшить шаг

def minimiz1(X,a,global\_epsilon,chislo): # для функции одной переменной

F = function3(X)

y = 0

fix = 0

gradient = 0

for k in range(len(X)):

gradient += derivative(X[k],k)

#print('gradient: ',gradient)

print('gradient: '.upper(),gradient)

X2 = X.copy()

for i in range(len(X)):

X2[i] = X[i] - a\*gradient

print('X2: ',X2)

F2 = function3(X2) # должны получить значение функции с параметрами из X2

print('F2(x2): ',F2)

print('F1(x2): ',F)

print('abs(F2-F): ',abs(F2-F),'\n')

chislo += 1

if abs(F2-F) <= global\_epsilon: # критерий останова

return (X2,F2,abs(F2-F),chislo) # решение, достигнутая точность, число итераций

else:

Res = minimiz1(X2,a,global\_epsilon,chislo)

return Res

#return (x2,abs(F2-F),chislo)

#return

# функция прорисовки графика для функции

def draw(point,X,N): # N - число разбиений по каждой оси

point\_x, point\_y, point\_z = point

#fig = plot.figure(figsize = (X[0], X[1])) # (figsize = (X\_old[0], X\_old[1]))

fig = plot.figure(figsize = (10, 10))

ax = fig.add\_subplot(1, 1, 1, projection = '3d') # размещаем оси на сетке

#xval = np.linspace(-X[0],X[1],N)

#yval = np.linspace(-X[0],X[1],N)

xval = np.linspace(-4,4,N)

yval = np.linspace(-4,4,N)

X, Y = np.meshgrid(xval, yval) # список координатных сеток

Resul = np.zeros(N\*N).reshape(N,N) # вместо 4,4 X[0],X[1]

X2 = np.zeros(2)

for i in range(len(X)):

for j in range(len(X)):

#print(i,j)

X2[0], X2[1] = X[i][j], Y[i][j]

#print(X2)

F = np.sin(X2[0]) + np.sin(X2[1]) # 1 тест

#F = X2[0]\*\*2 + X2[1]\*\*2 # 2 тест

#F = 0.26 \* (X2[0]\*\*2 + X2[1]\*\*2) - 0.48 \* X2[0] \* X2[1] # 3 тест(может стоит заменить)

#F = -20.0 \* np.exp(-0.2 \* np.sqrt(0.5 \* (X2[0]\*\*2 + X2[1]\*\*2))) - np.exp(0.5 \* (np.cos(2 \* np.pi \* X2[0]) + np.cos(2 \* np.pi \* X2[1]))) + np.e + 20

Resul[i][j] = F

F = 0

print('')

ax.scatter(point\_x, point\_y, point\_z, color='red')

surf = ax.plot\_surface(X, Y, Resul, rstride = 5,cstride = 5, cmap = cm.plasma)

plot.show()

return

#name\_test = input('Введите имя теста: ')

Resul = ()

Resul = minimiz1(X,a,global\_epsilon,chislo)

print('-------------------------------------------------')

# решение, достигнутая точность, число итераций

print('найденная точка: ',Resul[0])

print('значение в этой точке: ',Resul[1])

print('достигнутая точность: ',Resul[2])

print('число итераций: ',Resul[3])

print('шаг: ',a)

min\_x = Resul[0][0]

print(len(Resul))

point = []

for k in Resul[0]:

point.append(k)

point.append(Resul[1])

print(point)

if len(Resul[0]) == 1:

#minimum = (min\_x,Resul[1])

draw\_2d(X,100)

if len(Resul[0]) == 2:

min\_y = Resul[0][1]

#minimum = (min\_x,min\_y,Resul[1]) # может не нужно

draw(point,X,100)