|  |  |  |  |
| --- | --- | --- | --- |
| Министерство образования Республики Беларусь  Учреждение образования  Белорусский Государственный Университет Информатики и Радиоэлектроники | | | |
| Факультет | Компьютерных сетей и систем | | |
| Кафедра | Информатики  Дисциплина: Конструирование те технологии электронных вычислительных средств | | |
|  |  | | |
| **Отчет ПО Лабораторной Работе**  по курсу «Системы аналитического программирования» | | | |
| Студент:  гр. 758641  Ярош Г.И. | |  | Проверил:  Герман О.В. |
| Минск, 2018 | | | |

# ПОИСК МИНИМУМА ФУНКЦИИ С ПОМОЩЬЮ МЕТОДА ГРАДИЕНТНОГО СПУСКА

## Цель

Изучить возможности языка Python и библиотеки SciPy для поиска минимума функции. Найти минимум следующей функции:

## Теоретическая часть

Для поиска минимума функции подходит метод градиентного спуска. Он заключается в том, что на каждом шаге работы метода текущее значение минимума уменьшается на значение градиента функции в этой точке. Работа метода останавливается, когда значение градиента становится слишком маленьким.

В общем виде метод градиентного спуска для функции *n* переменных выглядит следующим образом:

(1)

Параметр определяет скорость градиентного спуска и устанавливается в значение меньше 1. Для исходной функции формула выше приобретает вид:

(2) (3)

Для получения минимума функции задаются значения . Затем итеративно производится расчет по формулам (1) и (2).

## Реализация

Для поиска минимума функции с помощью метода градиентного спуска в пакете SciPy в модуле optimize реализована функция fmin\_cg. На вход она принимает исходную функцию, градиент функции и значения . На выходе она возвращает минимум функции , а также массив точек пройденных в процессе работы алгоритма.

Исходный код программы приведен ниже:

*# -\*- coding: utf-8 -\*-*import numpy as np  
from matplotlib import pyplot as plt  
from mpl\_toolkits.mplot3d import axes3d as \_  
from scipy import optimize  
  
  
*# исходная функция*def f(x):  
 return 4 \* x[0] \*\* 2 + 2 \* x[0] \*\* 2 \* x[1] \*\* 2 + 4 \* x[1] \*\* 2 + 7  
  
  
*# градиент исходной функции*def gradf(x):  
 gx1 = 4 \* 2 \* x[0] + 2 \* x[0] \* x[1] \*\* 2  
 gx2 = 2 \* x[0] \*\* 2 \* x[1] + 2 \* 4 \* x[1]  
 return np.array([gx1, gx2])  
  
  
def main():  
 *# поиск минимума функции* x\_min, x\_grad = optimize.fmin\_cg(f, np.array([4.5, 4.6]), fprime=gradf, retall=True)  
  
 *# инициализация значений для графиков* interval = np.arange(-5, 5, 0.1)  
 x = np.array([[e] \* len(interval) for e in interval])  
 y = np.array([[e for e in interval]] \* len(interval))  
 z = np.array([[f((u, v)) for u in interval] for v in interval])  
  
 *# график поверхности функции* fig = plt.figure()  
 ax = fig.add\_subplot(111, projection='3d')  
 ax.plot\_wireframe(x, y, z, color=(0, 0, 0.8, 0.2), label='f(u, v)')  
  
 *# график изменения градиента функции* ax.plot(  
 [x[0] for x in x\_grad],  
 [x[1] for x in x\_grad],  
 [f(x) for x in x\_grad],  
 '--.',  
 color=(0.5, 0, 0, 1), linewidth=1, markersize=4, label='grad(f(u, v))'  
 )  
  
 *# начальная точка и точка минимума функции* ax.scatter(  
 x\_grad[0][0], x\_grad[0][1], f(x\_grad[0]), 'o',  
 color='blue', label='(x0, y0)', linewidths=4  
 )  
 ax.scatter(  
 x\_min[0], x\_min[0], f(x\_min), 'o',  
 color='green', label='(xmin, ymin)', linewidths=4  
 )  
  
 *# вывод графиков на экран и сохранение в файл* ax.legend(loc='lower center')  
 plt.show()  
 fig.savefig('plot.png')  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Полученные графики приведены на рисунке 1. Красной линией на графике изображено изменение градиента в процессе работы алгоритма.

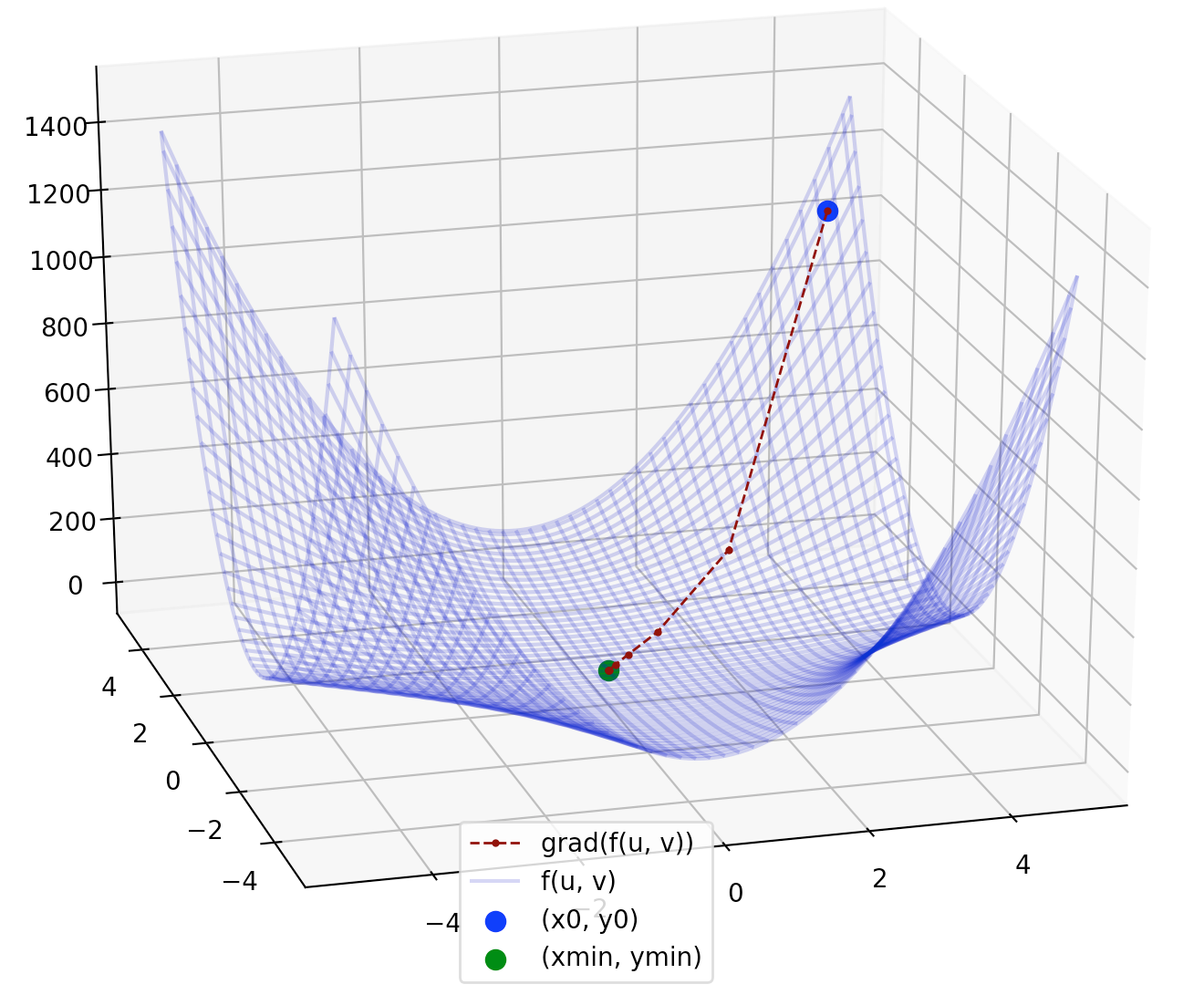


Рис. 1. График поверхности функции, изменения градиента, начальной точки и минимума функции.