Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра прикладной математики

Лабораторная работа № 4

РЕШЕНИЕ СИСТЕМ НЕЛИНЕЙНЫХ УРАВНЕНИЙ МЕТОДОМ НЬЮТОНА

ФАКУЛЬТЕТ:	ПМИ
ГРУППА:	ПМ-71
СТУДЕНТЫ:	Баштовой П.А. Востриков В.А.
ПРЕПОДАВАТЕЛЬ:	Задорожный А.Г. Патрушев И.И.

Новосибирск 2020

1) Условие задачи:

Графическая реализация результатов решения систем нелинейных уравнений методом Ньютона.

2) Структура программы:

Для создания программы-визуализатора была использована библиотека ***python matplotlib.***

3) Ход работы:

- Настройка функции:

```
def plot_function():

    x = np.arange(X_min, X_max, dx)

    # Сама функция
    y= x * x + 1
    plt.plot(x, y)
```

- Настройка градиента:

```
def plot_gradient():

    # Построение сетки
    y, x = np.mgrid[slice(X_min, X_max + dy, dy), slice(Y_min, Y_max +
dx, dx)]

    # Норма F
    z = norm(([y + x, y - x]))

    plt.contourf(x, y, z, 100, cmap='inferno')
    plt.colorbar()
```

- Подсчет нормы x:

```
def norm(x):
    sum = 0
    for values in x:
        sum = sum + values*values
    return np.sqrt(sum)
```

- Чтение из файлов и добавление графиков:

```
def plot_curve(x, y):
    f1 = open(x, 'r')
    f2 = open(y, 'r')
    string_x = f1.read()
    string_y = f2.read()
    x_ = string_x.split(' ')
    y_ = string_y.split(' ')
    plt.plot(x_, y_)

def plot_circle(x):
```

```
        f1 = open(x, 'r') # x, y, r
        string = f1.read()
        values = string.split(' ')
        C   =   matp.patches.Circle((values[0],   values[1]),   values[2],
    color='b', fill = False)
        plt.gcf().gca().add_artist(C)
```

4) *Входные данные:*

Файл «circle_info.txt»:

3 3 1

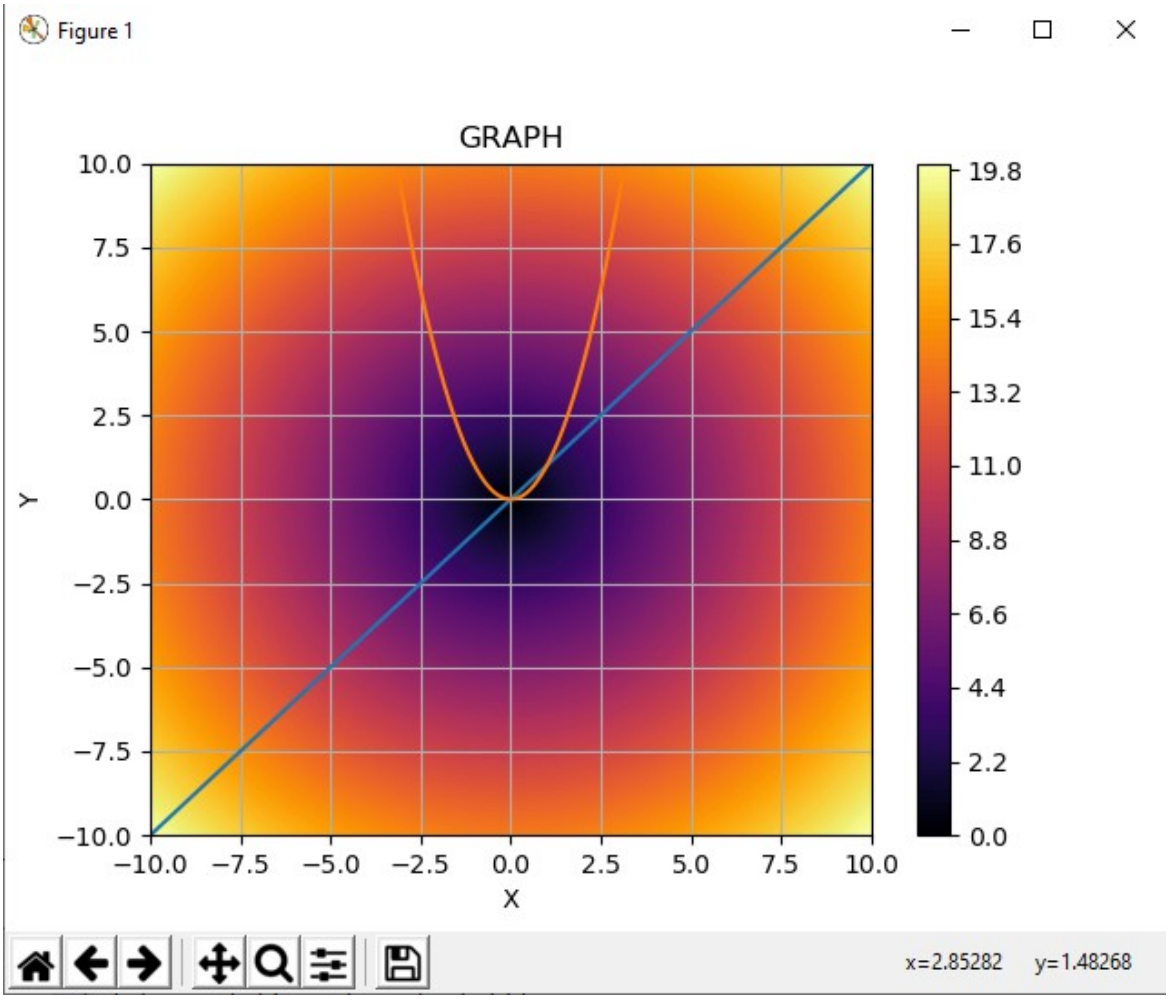
Файл «curve_x.txt»:

5 2 -1 4 5 2 7

Файл «curve_y.txt»:

3 2 1 4 5 6 7

5) *Результат работы программы:*



6) *Текст программы:*

```
from math import radians
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as matp


# Настройка осей
X_min = - 10
X_max = 10

Y_min = - 10
Y_max = 10


# Настройка разбиения
dx = 0.01
dy = 0.01


# В файлах указывать массивы значений x, y. В аргументах указывать адрес соответствующих массивов
def plot_curve(x, y):
    f1 = open(x, 'r')
    f2 = open(y, 'r')
    string_x = f1.read()
    string_y = f2.read()
    x_ = string_x.split(' ')
    y_ = string_y.split(' ')
    plt.plot(x_, y_)



# В файле указывать три значения x, y, r. В аргументе указывать адрес
def plot_circle(x):
    f1 = open(x, 'r') # x, y, r
    string = f1.read()
    values = string.split(' ')
    C = matp.patches.Circle((values[0], values[1]), values[2], color='b', fill = False)
    plt.gcf().gca().add_artist(C)



def norm(x):
    sum = 0
```

```
    for values in x:
        sum = sum + values*values
    return np.sqrt(sum)



# Ручная установка!
def plot_gradient():

    # Построение сетки
    y, x = np.mgrid[slice(X_min, X_max + dy, dy), slice(Y_min, Y_max + dx, dx)]

    # Норма F
    z = norm(([y + x, y - x]))

    plt.contourf(x, y, z, 100, cmap='inferno')
    plt.colorbar()


# Ручная установка!
def plot_function():

    x = np.arange(X_min, X_max, dx)

    # Сама функция
    y= x * x + 1
    plt.plot(x, y)


# Ручная установка!
def main():

    plt.axis([X_min, X_max, Y_min, Y_max])

    plt.xlabel('X')
    plt.ylabel('Y')
    plt.title('GRAPH')
    plt.grid(True)

####

#plot_curve('curve_x.txt', 'curve_y.txt')
#plot_circle('circle_info.txt')

plot_gradient()
plot_function()
main()
plt.show()
####
```