# Министерство образования и науки Российской Федерации

## Федеральное государственное бюджетное образовательное учреждение высшего образования

## «Новосибирский государственный технический университет»

NSTU_Logo_blue

Кафедра прикладной математики

|  |
| --- |
| Лабораторная работа № 4 |

РЕШЕНИЕ СИСТЕМ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

МЕТОДОМ НЬЮТОНА

|  |  |  |
| --- | --- | --- |
| сигма градиент синий1 | 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.

1. ***Структура программы:***

Для создания программы-визуализатора была использована библиотека

***python matplotlib.***

1. ***Ход работы:***

* Настройка функции:

def plot\_function():

x = np.arange(X\_min, X\_max, dx)

# функция

y= x \* x

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)

1. ***Входные данные:***

Файл «circle\_info.txt»:

3 3 1

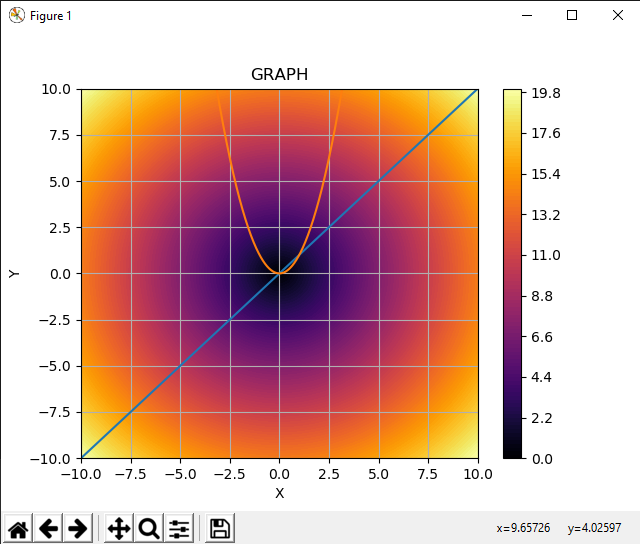
Файл «curve\_x.txt»:

5 2 -1 4 5 2 7

Файл «curve\_y.txt»:

3 2 1 4 5 6 7

1. ***Результат работы программы:***



1. ***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

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()

####