**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №1**

**по дисциплине «Информатика»**

Тема: Основные управляющие конструкции языка Python

|  |  |  |
| --- | --- | --- |
| Студент гр. 3342 |  | Корниенко А.Е. |
| Преподаватель |  | Иванов Д. В. |

Санкт-Петербург

2023

## Цель работы

Научиться работать с библиотекой numpy языка Python, изучить функции для работы с матрицами. С их помощью реализовать три функции для работы с матрицами и осуществить линейные операции над ними.

## Задание

Задача 1.

Оформите решение в виде отдельной функции check\_collision. На вход функции подаются два ndarray -- коэффициенты bot1, bot2 уравнений прямых bot1 = (a1, b1, c1), bot2 = (a2, b2, c2) (уравнение прямой имеет вид ax+by+c=0).

Функция должна возвращать точку пересечения траекторий (кортеж из 2 значений), предварительно округлив координаты до 2 знаков после запятой с помощью round(value, 2).

Задача 2.

Оформите задачу как отдельную функцию check\_surface, на вход которой передаются координаты 3 точек (3 ndarray 1 на 3): point1, point2, point3. Функция должна возвращать коэффициенты a, b, с в виде ndarray для уравнения плоскости вида ax+by+c=z. Перед возвращением результата выполнение округление каждого коэффициента до 2 знаков после запятой с помощью round(value, 2).

Задача 3.

Оформите задачу как отдельную функцию check\_surface, на вход которой передаются координаты 3 точек (3 ndarray 1 на 3): point1, point2, point3. Функция должна возвращать коэффициенты a, b, с в виде ndarray для уравнения плоскости вида ax+by+c=z. Перед возвращением результата выполнение округление каждого коэффициента до 2 знаков после запятой с помощью round(value, 2).

## Выполнение работы

Для работы с линейными операциями, массивами и матрицами была использована библиотека *numpy*.

1. *def check\_collision(bot1, bot2)*: Необходимо найти точку пересечения двух прямых. Функция принимает на вход коэффициенты уравнений прямых. Функция решает систему линейных уравнений *linalg.solve*. Возвращает кортеж – решение системы уравнений.
2. *def check\_surface(point1, point2, point3)*: Аргументы функции – списки, в которых содержатся координаты точек. При помощи *linalg.solve* метода библиотеки *numpy* решаем состему из трех уравнений и находим коэффициенты уравнения. Перед этим нужно проверить существование решения системы методом *linalg.matrix\_rank*. Возвращает массив – коэфициенты c.
3. *def check\_rotation(vec, rad)*: Необходимо найти координаты матрицы, повернутой на угол *rad*. Составили формулу матрицы поворота (см. рис. 1). Результатом будет умножение этой матрицы на исходную. Возвращает массив – матрица, полученная после умножения.

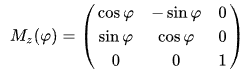


Рисунок 1 - Матрица поворота вокруг оси z

Переменные:

* a1, a2, a3, b1, b2, b3, с1, с2, с3 – элементы, на которые разбиваются переданные в качестве аргумента массивы
* Списки: mat\_n, arr\_1, arr\_2, res1, res2.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
| 1. | [- 3, -6, 9]),[8, -7, 0] | (0.91, 1.04) |  |
| 2. | [ 1, -6, 1]), [ 0, -3,2], [3, 0, -1] | [2. 1. 5.] |  |
| 3. | [ 1, -2, 3], 1.57 | [2. 1. 3.] |  |

## Выводы

Были изучены конструкции языка Python по работе с библиотекой numpy.

Разработаны функции для обработки матриц и решений линейных уравнений.

# Приложение А Исходный код программы

Название файла: main.py

import numpy as np

def check\_collision(bot1, bot2):

mat\_n = []

mat\_n.append(bot1[:2])

mat\_n.append(bot2[:2])

mat = np.array(mat\_n)

p = []

if np.linalg.matrix\_rank(mat) < 2:

return None

x = np.linalg.solve(mat, -np.array([bot1[2], bot2[2]]))

for i in x:

p.append(round(i, 2))

return tuple(p)

def check\_surface(point1, point2, point3):

a1, b1, с1 = [int(i) for i in point1]

a2, b2, с2 = [int(i) for i in point2]

a3, b3, с3 = [int(i) for i in point3]

arr\_1 = np.array([[a1, b1, 1], [a2, b2, 1], [a3, b3, 1]])

arr\_2 = np.array([с1, с2, с3])

if np.linalg.matrix\_rank(arr\_1) < 3:

return None

result1 = np.linalg.solve(arr\_1, arr\_2)

r = []

for i in result1:

r.append(round(i,2))

result = np.array(r)

return result

def check\_rotation(vec, rad):

mat = np.array(vec)

matrix\_pov = np.array([[np.cos(rad), np.sin(rad), 0],

[-np.sin(rad), np.cos(rad), 0],

[0, 0, 1]])

res1 = mat.dot(matrix\_pov)

res2 = []

for i in res1:

res2.append(round(i, 2))

res = np.array(res2)

return res