МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

“**ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**”

Проект второго курса

Отчёт

Направление подготовки Прикладная математика и информатика

Направленность программы 01.03.02 Прикладная математика и информатика

­­­­­­­­­­

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Куничкин Б.В.

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Заброда А. В.

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Куштин И. В.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ О. Г. Пустовалова

Ростов-на-Дону 2022

Отчёт

Куничкин Б.В, Куштин И.В, Заброда А.В.

31 мая 2022 г.

1. **Первое задание**
   1. **Постановка задачи**

Сформировать 50 файлов, в каждый из которых записать матрицу 100x100 из случайных целых чисел в диапазоне от -100 до 100. Для матриц сравнить время работы алгоритмов, реализованных в MATLAB и Python (NumPy). Результаты записать во вновь созданные 50 файлов. Оформить отчёт в Microsoft Word.

* 1. **Код решения MATLAB**

clc, clear

filetime = fopen('results\_time.txt','w'); %открытие файла для результатов времени выполнения алгоритма

for i = 1:50

filename1 = strcat('matrixs/matrix', num2str(i), '.txt');

slau = randi([-100, 100], 100, 101); %создаем матрицу в которой содержатся

% матрица порядка 100 и вектор-столбец свободных членов

writematrix(slau,filename1); % запись в файл матрицы и вектора свободных членов

slau1 = readmatrix(filename1);

tic; % начало отсчета времени выполнения алгоритма

A = slau(:,1:5); % матрица А

b = slau(:,6); % вектор свободных членов

x=b\A; % решение СЛАУ

time\_end = toc; % конец отсчета времени выполнения алгоритма

filename2 = strcat('results/result', num2str(i), '.txt');

writematrix(x,filename2); % запись в файл результата вычислений

fprintf(filetime, "Матрица %3d: вычислено за %12.8f секунд\n", i, time\_end); % запись

%результатов работы алгоритма

end

fclose(filetime);

* 1. **Код решения Python**

import numpy as np

import os

from time import \*

def timer\_func(func):

def wrap\_func(\*args, \*\*kwargs):

t1 = time() # время на начало выполнения функции

result = func(\*args, \*\*kwargs)

t2 = time() # время на конец выполнения функции

with open("time.txt", mode='a') as f:

f.write(f"{(t2-t1):.4f}s\n")

print(f"{(t2-t1):.4f}s\n")

return result

return wrap\_func

@timer\_func

def matrix\_solve(m):

A = m[:, :-1] # матрица

b = m[:, -1] # вектор свободных членов

x = np.linalg.solve(A, b)

return x

if \_\_name\_\_ == '\_\_main\_\_':

os.makedirs("data", exist\_ok=True)

for i in range(50):

# создаём матрицу со 100 строками и 101 столбцом, где последний столбец будем в будущем использовать как вектор свободных членов

m = np.random.randint(-100, 101, size=(100, 101))

# сохраняем в файл матрицу

np.savetxt(f"data/np\_matrix\_{i}.txt", m, fmt="%i")

# получаем из файла матрицу (на самом деле это лишний шаг, можно его убрать и заменить m1 на m)

m1 = np.loadtxt(f"data/np\_matrix\_{i}.txt", usecols=range(101), dtype=int)

# решаем матрицу и сохраняем в файл результат

x = matrix\_solve(m1)

np.savetxt(f"data/np\_matrix\_solved\_{i}.txt", x, fmt="%f")

* 1. **Результаты работы**

**Python**

Среднее время выполнения одной итерации вычислений на Python заняло 0.004 секунды.

**MATLAB**

Среднее время выполнения одной итерации вычислений в MATLAB заняло 0.00006 секунды.

* 1. **Вывод**

При решении данной задачи мы выяснили, что в среднем Python затрачивает на 0,00394 секунду больше, чем MATLAB. При работе с матрицами намного удобнее использовать MATLAB, т. к. программисту требуется меньше трудо- и времязатрат на написание кода в MATLAB.

1. **Второе задание**
   1. **Постановка задачи:**

Вычисления выполнять в MATLAB и Python (SymPy). Вычислить неопределённый интеграл.

В цикле вычислить определенные интегралы для ста вариантов верхнего и нижнего пределов (пределы и константы (a, b, … ) задать самостоятельно). Результаты записать в текстовый файл. Сравнить результаты вычисления и время вычислений в MATLAB и Python. Оформить отчёт в Microsoft Word.

* 1. **Код решения MATLAB**

clc, clear

syms x a b

filename = fopen('integrals\_matlab.txt','w'); % открываем файл на запись

fprintf(filename,'%9s%11s%11s%13s%13s%24s\n','Nomer','lim1', 'lim2', 'a', 'b', 'result');

% печатаем шапку таблицы

a = 54; % задаем константу

b = 77; % задаем константу

lim1 = 2; % задаем предел интегрирования для отчета

lim2 = 10; % задаем предел интегрирования для отчета

integ = int(1 / x / sqrt(a\*x+b), x); % вычисляем неопределенный интеграл

result = subs(integ, x, lim2) - subs(integ, x, lim1); % результат для отчета

fprintf(filename,'%9d%11d%11d%13d%13d%24.6f\n', 1, lim1, lim2, a, b, result); % выводим переменные

tic; % начало отсчета времени выполнения алгоритма

for i = 2:100

lim1 = randi([1, 100]); % рандомное число

lim2 = randi([1, 100]); % рандомное число

result = subs(integ, x, lim2) - subs(integ, x, lim1); % подставляем пределы интегрирования

fprintf(filename,'%9d%11d%11d%13d%13d%24.6f\n', i, lim1, lim2, a, b, result); % выводим переменные

end

time\_end = toc; % конец отсчета времени выполнения алгоритма

fprintf(filename, "Вычислено за " + num2str(time\_end) + " секунд"); % печатаем время выполнения

fclose(filename); % закрываем файл

* 1. **Код решения Python**

from sympy import \*

from sympy.abc import a, x, y

import random

from time import \*

def rand\_integral(x, predel\_1, predel\_2):

result = (integ.subs(x, predel\_2) - integ.subs(x, predel\_1)).evalf() # результат вычисляем

print(result)

return (str(result) + '\n') # возвращаем результат в виде строки

if \_\_name\_\_ == '\_\_main\_\_':

t1 = time()

x = symbols("x")

s = str()

a = 54

b = 77

integ = integrate(1/x/sqrt(a\*x+b), x) # вычисляем неопределённый интеграл

predel\_1 = 2 # 1ый определённый интеграл у нас совпадает

predel\_2 = 10

s += rand\_integral(x, predel\_1, predel\_2)

for i in range(1, 100):

predel\_1 = random.randint(1, 100) # на каждой итерации подставляем новые пределы

predel\_2 = random.randint(100, 200)

s += rand\_integral(x, predel\_1, predel\_2)

with open("results.txt", mode="w") as f:

f.write(s) # записываем строку в файл

t2 = time()

print(f"{(t2-t1):.4f}s")

* 1. **Комментарии к решению**

1. При работе над нашей программой мы обнаружили, что как в MATLAB, так и в Python, интеграл не вычислялся. Мы решили эту проблему, заменив степень 0.5 на квадратный корень.
2. Все условия в задании генерируются случайным образом, кроме первого. Это нужно, чтобы проверить программы на равенство выдаваемого значения.
   1. **Результаты времени работы**

**Python**

Время выполнения вычислений на Python заняло 1.34 секунды.

**MATLAB**

Время выполнения вычислений в MATLAB заняло 1.33 секунды.

* 1. **Результаты работы**

**Python**

0.0906605026297954

**MATLAB**

0.090661

* 1. **Вывод**

В реализациях в MATLAB и Python предел интегрирования расставлялись случайным образом везде, кроме первого случая, следовательно его и будем сравнивать. Результаты получились одинаковые, но в MATLAB они более округлёнными.

При решении данной задачи мы выяснили, что в Python и MATLAB затрачивают одинаковое количество времени. Для программиста, работающего с интегралами, одинаково удобен как Python, так и MATLAB.

**Документация и учебники**

1. [**https://www.sympy.org/ru/**](https://www.sympy.org/ru/)
2. [**https://live.sympy.org/**](https://live.sympy.org/)