МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

**«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Институт математики, механики и компьютерных наук им. И.И.Воровича Кафедра математического моделирования

Проект второго курса

Отчет

Направление подготовки Прикладная математика и информатика Направленность программы 01.03.02 Прикладная математика и информатика

|  |  |  |
| --- | --- | --- |
| Обучающийся |  | C.С. Сальников |
| Обучающийся |  | Д.Н. Дурицкий |
| Обучающийся |  | А.М. Меркулов |
| Обучающийся |  | Д.В. Радионов |
| Руководитель |  | О.Г. Пустовалова |

Ростов-на-Дону 2022

**Вариант 3**

**1 Задание NumPy**

**1.1 Постановка задачи**

Сформировать 50 файлов, в каждый из которых записать матрицу 100х100 из случайных целых чисел в диапазоне от -100 до 100. Для матриц сравнить время работы алгоритмов(согласно номеру варианта), реализованных в MATLAB и Python (Numpay).Результаты записать во вновь созданные 50 файлов.Оформить отчет в MicrosoftWord.

Алгоритм - Вектор уникальных значений в массиве.

**1.2 Код Matlab**

clc

clear

fID = fopen("indreznumpy.txt", 'w'); % Файл со всеми временами выполнения для всех матриц

for i = 1:50

A = randi([-100 100], 100);

fileID = strcat('ind', num2str(i), '.txt');

writematrix(A, fileID, 'Delimiter', '\t');

tic

A1 = readmatrix(fileID);

x = unique(A1);

fid = fopen("numpyrez"+string(i)+".txt", "w"); % Файл со результатом для одной матрицы

fprintf(fid, '%f\n', x);

fclose(fid);

a = toc

fprintf(fID, string(i)+'. %f\n', a);

end

fclose(fID);

**1.3 Код Python**

import numpy as np  
from datetime import datetime  
import time  
f = open('indreznumpy\_py.txt', 'w')  
for i in range(1, 51):  
start\_time = [datetime.now](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fdatetime.now)()  
a = np.genfromtxt('ind' + str(i) + '.txt', delimiter=' ')  
res = np.unique(a)  
np.savetxt('numpyrez' + str(i) + '\_py.txt', res, fmt = '%11.6f')  
f.write(str(i)+'. '+str([datetime.now](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fdatetime.now)()-start\_time)[6:: + '\n')  
f.close()

**1.4 Сравнение результатов**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | MatLab | | 1. 0.009394 | | 2. 0.036336 | | 3. 0.009571 | | 4. 0.017512 | | 5. 0.009125 | | 6. 0.009848 | | 7. 0.008568 | | 8. 0.016139 | | 9. 0.008052 | | 10. 0.008322 | | 11. 0.008175 | | 12. 0.008553 | | 13. 0.009016 | | 14. 0.008865 | | 15. 0.008185 | | 16. 0.008170 | | 17. 0.008566 | | 18. 0.009147 | | 19. 0.008056 | | 20. 0.008103 | | 21. 0.008246 | | 22. 0.008457 | | 23. 0.008235 | | 24. 0.008386 | | 25. 0.008801 | | 26. 0.010377 | | 27. 0.009454 | | 28. 0.008592 | | 29. 0.008843 | | 30. 0.007840 | | 31. 0.008192 | | 32. 0.008555 | | 33. 0.008150 | | 34. 0.008108 | | 35. 0.008694 | | 36. 0.008039 | | 37. 0.010382 | | 38. 0.008446 | | 39. 0.007981 | | 40. 0.008553 | | 41. 0.007933 | | 42. 0.008771 | | 43. 0.008182 | | 44. 0.008878 | | 45. 0.011119 | | 46. 0.007810 | | 47. 0.008338 | | 48. 0.008259 | | 49. 0.016114 | | 50. 0.008012 | | |  | | --- | | NumPy | | 1. 0.002002 | | 2. 0.024994 | | 3. 0.001001 | | 4. 0.002001 | | 5. 0.001001 | | 6. 0.002002 | | 7. 0.002002 | | 8. 0.001001 | | 9. 0.002002 | | 10. 0.001001 | | 11. 0.001001 | | 12. 0.002002 | | 13. 0.001000 | | 14. 0.001002 | | 15. 0.002001 | | 16. 0.001002 | | 17. 0.001000 | | 18. 0.002002 | | 19. 0.008011 | | 20. 0.001997 | | 21. 0.001002 | | 22. 0.002001 | | 23. 0.008008 | | 24. 0.001001 | | 25. 0.002002 | | 26. 0.001001 | | 27. 0.001000 | | 28. 0.002002 | | 29. 0.001001 | | 30. 0.001001 | | 31. 0.002002 | | 32. 0.001001 | | 33. 0.002002 | | 34. 0.001000 | | 35. 0.002003 | | 36. 0.000999 | | 37. 0.001002 | | 38. 0.003004 | | 39. 0.001000 | | 40. 0.002002 | | 41. 0.001000 | | 42. 0.001001 | | 43. 0.002003 | | 44. 0.001000 | | 45. 0.002002 | | 46. 0.001002 | | 47. 0.001000 | | 48. 0.002001 | | 49. 0.001001 | | 50. 0.002002 | |

Суммарное время MatLab: 0.438 сек.  
Суммарное время Python Numpy: 0.109 сек.

**1.5 Вывод:**

По итогам работ можем сделать вывод о том, что Python Numpy справляется с работой с случайно генерируемыми матрицами в ~4 раза быстрее, а также более прост в использовании чем MatLab.

**2 Задание SymPy**

**2.1 Постановка задачи**

Вычисления выполнять в MATLAB и Python(Sumpay).Вычислить неопределенныйинтеграл.В цикле вычислить определенныеинтегралыдля ста вариантов верхнего и нижнего пределов(пределы и константы(a,b,...)задатьсамостоятельно).Результаты записать втекстовыйфайл.Сравнить результаты вычисленияи время вычисленийвMATLAB и Python. Подынтегральная функция 1/(a\*x+b).

**2.2 Код Matlab**

clc

clear;

syms x

a = 1

b = 1

f = 1 / (a \* x + b)

int(f, x)

lim1 = linspace(1, 100, 100);

lim2 = linspace(1, 100, 100);

lim2 = lim2.^2

fid = fopen("number\_2.txt", "w")

fid2 = fopen("timeNumber2.txt", "w")

tic

for i = 1:100

expr = string(eval(int(f, [lim1(i), lim2(i)]))) + newline;

fwrite(fid, expr);

end

t = toc

fwrite(fid2, string(t));

fclose(fid)

fclose(fid2)

**2.3 Код Python**

import sympy as sp  
import random  
import time  
  
timing = time.time()  
a = int(input("a » "))  
b = int(input("b » "))  
  
while (a == 0 and b == 0):  
a = int(input("a » "))  
b = int(input("b » "))  
  
lim1 = [random.randint(1, 1000) for i in range(100)]  
lim2 = [random.randint(1, 1000) for i in range(100)]  
  
x = sp.Symbol("x")  
  
f = open('number\_3.txt', 'w')  
[#print](https://vk.com/im?sel=347255471&st=%23print)(lim1)  
[#print](https://vk.com/im?sel=347255471&st=%23print)(lim2)  
for i in range(100):  
if(a != 0):  
ans = sp.integrate(1 / (a \* x + b), (x, lim1[i], lim2[i]))  
else:  
ans = b \* lim2[i] - lim1[i] \* b  
f.write(str(ans) + '\n')  
  
f.write(str(time.time() - timing) + " seconds. Programm end")  
f.close()

**2.4 Сравнение результатов:**

Cуммарное время MatLab: 0.53822 cек.  
Суммарное время Python Sympy: 1.21507 cек.

Результаты вычислений идентичны, ввиду специфики задания (пределы интегрирования и коэффициенты многочленов задаются одинаковые).

**2.4 Вывод**

По результатам работы программ можем сделать вывод о том, что среда MatLab не только более легка в использовании для вычисления интегралов, но и работает в ~2.4 раза быстрее Python Sympy.

**3 Литература:**

1. Руководство пользователя Numpy - <https://numpy.org/doc/stable/user/index.html> [14.05.2022]

2. Документация Numpy - [https://numpy.org/doc/stable/reference/index.html#reference](https://numpy.org/doc/stable/reference/index.html%23reference) [14.05.2022]

3. Документация Sympy - <https://www.sympy.org/ru/> [23.05.2022]

4. Документация по интегралам(Sympy) –   
<https://docs.sympy.org/latest/modules/integrals/integrals.html> [23.05.2022]

5. Документация по интегралам(Численные методы Sympy) - <https://docs.sympy.org/latest/modules/integrals/integrals.html#numeric-integrals> [23.05.2022]