

**Федеральное государственное автономное образовательное
учреждение высшего образования
«Национальный исследовательский университет
«Высшая школа экономики»**

Образовательная программа
«Прикладная математика»

**ОТЧЕТ
по лабораторной работе № 1**

По теме
«Теория погрешностей и машинная арифметика»

Выполнил
студент группы БПМ211
Кудряшов Максим Дмитриевич

Проверил
Брандышев Петр Евгеньевич

Москва - 2024

Содержание

1	Расчет погрешности частичных сумм ряда (№ 1.1.8)	2
1.1	Формулировка	2
1.2	Найдем сумму ряда аналитически	2
1.3	Найдем частичные суммы ряда и их погрешности	2
1.3.1	Код на Python	2
1.4	Посчитанные данные	4
2	Расчет погрешности матрицы (№ 1.9.2)	4
2.1	Формулировка	4
2.2	Теория	4
2.3	Код на Python	4
2.4	Вывод программы	5
2.5	Выводы	5
3	Нахождение машинного нуля (№ 1.6, 1.7)	6
3.1	Формулировка	6
3.2	Код на Python	6
3.3	Код на C++	7
3.4	Вывод кода на Python	8
3.5	Вывод кода на C++	8
3.6	Сравнение результатов	8

1 Расчет погрешности частичных сумм ряда (№ 1.1.8)

1.1 Формулировка

1. Найти сумму ряда S аналитически.

$$S = \sum_{n=0}^{\infty} \frac{32}{n^2 + 9n + 20}$$

2. Найти частичные суммы ряда S_N при $N = 10, 10^2, 10^3, 10^4, 10^5$.

$$S_N = \sum_{n=0}^N \frac{32}{n^2 + 9n + 20}$$

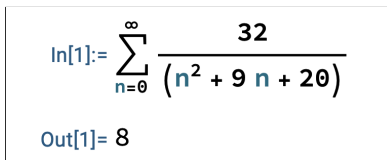
3. Вычислить погрешности для каждого N .

4. Вычислить количество значащих цифр для каждого N .

5. Построить гистограмму зависимости верных цифр результата от N .

1.2 Найдем сумму ряда аналитически

Посчитаем значение предела в Wolfram:



1.3 Найдем частичные суммы ряда и их погрешности

Для различных значений N вычислим частичные суммы, погрешности и количество значащих цифр.

1.3.1 Код на Python

```

1  from collections import defaultdict
2  from prettytable import *
3
4
5  def calculateNum(n: int):
6      return 32 / (n ** 2 + 9 * n + 20)
7
8
9  def calculateSum(N: int):
10     s = 0
11     for n in range(N):
12         s += calculateNum(n)
13     return s
14
15
16  def calculateData():

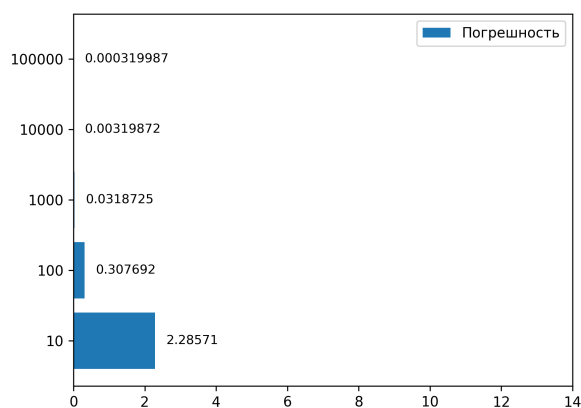
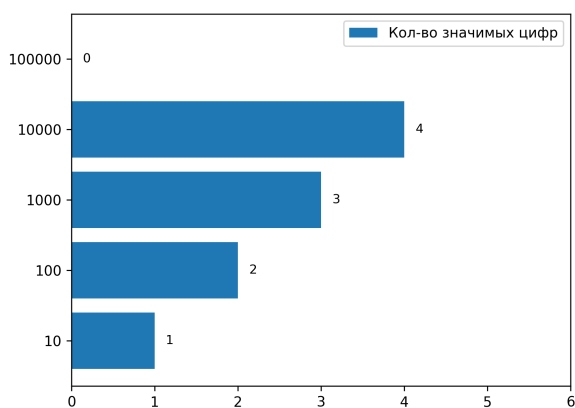
```

1.3 Найдем частичные суммы ряда и погрешности ЧАСТИЧНЫХ СУММ РЯДА (№ 1.1.8)

```
17 N_list = [10 ** k for k in range(1, 6)]
18
19 sums = {}
20 errors = {}
21 digits = defaultdict(int)
22 exact_sum = 8
23
24 for N in N_list:
25     sums[N] = calculateSum(N)
26     errors[N] = abs(sums[N] - exact_sum)
27
28 for N in N_list:
29     i = -1
30     for i in range(0, 50):
31         order_of_error = 1 / (10 ** i)
32         if errors[N] <= order_of_error:
33             digits[N] += 1
34         else:
35             break
36
37 # for N in N_list:
38 #     print(f"{sums[N]}\t#\t{errors[N]}\t{digits[N]} значащих цифр, {N = }")
39
40 table = PrettyTable()
41 table.field_names = ["N", "Значение", "Погрешность", "Число значащих цифр"]
42 table.align = 'l'
43 for N in N_list:
44     table.add_row([N, sums[N], errors[N], digits[N]])
45 print(table)
46
47 return {
48     "sums": sums,
49     "errors": errors,
50     "digits": digits
51 }
```

1.4 Посчитанные данные

N	Значение	Погрешность	Число значащих цифр
10	5.714285714285714	2.2857142857142856	0
100	7.69230769230769	0.30769230769231015	1
1000	7.968127490039844	0.03187250996015578	2
10000	7.9968012794882295	0.003198720511770503	3
100000	7.999680012799457	0.00031998720054282614	4



2 Расчет погрешности матрицы (№ 1.9.2)

2.1 Формулировка

Для матрицы A решить вопрос о существовании обратной матрицы в следующих случаях:

1. элементы матрицы заданы точно;
2. элементы матрицы заданы приближенно с относительной погрешностью $\alpha = 0.05$ и $\beta = 0.1$

$$A = \begin{pmatrix} 30 & 34 & 19 \\ 31.4 & 35.4 & 20 \\ 24 & 28 & 13 \end{pmatrix}$$

2.2 Теория

ДОПИСАТЬ ТЕОРИЮ, СКАЗАТЬ О ТЕОРЕМЕ

2.3 Код на Python

```

1 import numpy as np
2 from itertools import product
3
4
```

```

5  def is_inverse_matrix_exist(matrix: np.ndarray):
6      matrix_det = np.linalg.det(matrix)
7      print(f'Определитель без погрешности {matrix_det}')
8      print()
9
10
11 def is_inverse_matrix_exist_with_error(matrix: np.ndarray, delta: float):
12     metrix_dets = []
13     for sign in list(product([-1, 1], repeat=9)):
14         new_matrix = matrix * (1 + np.array(sign).reshape(3, 3) * delta)
15         metrix_dets.append(np.linalg.det(new_matrix))
16
17     min_det = np.min(metrix_dets)
18     max_det = np.max(metrix_dets)
19     print(f'Минимальное значение определителя = {min_det}')
20     print(f'Максимальное значение определителя = {max_det}')
21
22     if min_det < 0 < max_det:
23         print(f'При относительной погрешности {delta} определитель может обратиться в 0")
24     else:
25         print(f'При относительной погрешности {delta} определитель не может обратиться в 0")
26     print()
27
28
29 matrix = np.array([[30, 34, 19],
30                    [31.4, 35.4, 20],
31                    [24, 28, 13]])
32
33 is_inverse_matrix_exist(matrix)
34 is_inverse_matrix_exist_with_error(matrix, 0.05)
35 is_inverse_matrix_exist_with_error(matrix, 0.1)

```

2.4 Вывод программы

Определитель без погрешности 9.6000000000000069

Минимальное значение определителя = -984.87280000000016

Максимальное значение определителя = 1027.89900000000008

При относительной погрешности 0.05 определитель может обратиться в 0

Минимальное значение определителя = -2965.2384

Максимальное значение определителя = 3032.29599999999985

При относительной погрешности 0.1 определитель может обратиться в 0

2.5 Выводы

Если значения заданы точно, то определитель не равен 0, а следовательно существует обратная матрица.

Если элементы матрицы заданы приближенно с относительной погрешностью $\alpha = 0.05$ и $\beta = 0.1$, то определитель можно равняться нулю, значит обратная матрица может не существовать.

3 Нахождение машинного нуля (№ 1.6, 1.7)

3.1 Формулировка

Вычислить значения машинного нуля, машинной бесконечности, машинного эпсилон в режимах одинарной, двойной и расширенной точности на двух алгоритмических языках. Сравнить результаты.

3.2 Код на Python

```

1  import numpy as np
2
3
4  def print_zero(my_type):
5      k = 0
6      num = my_type(1)
7      while num != 0:
8          num = my_type(num / 2)
9          k += 1
10     print(my_type, f"машинный ноль = 2^{-k}")
11
12
13  def print_infinity(my_type):
14      k = 0
15      num = my_type(1)
16      while num != np.inf:
17          num = my_type(num * 2)
18          k += 1
19     print(my_type, f"машинная бесконечность = 2^{k}")
20
21
22  def print_epsilon(my_type):
23      k = 0
24      num = my_type(1)
25      while my_type(1.) + num > my_type(1.):
26          num = my_type(num / 2)
27          k += 1
28     print(my_type, f"машинное эпсилон = 2^{-k}")
29
30
31  for my_type in [np.single, np.double, np.longdouble]:
32     print_zero(my_type)
33     print_infinity(my_type)
34     print_epsilon(my_type)
35     print()

```

3.3 Код на C++

```
1  #include <iostream>
2  #include <cmath>
3
4  template <typename T>
5  void print_zero() {
6      int k = 0;
7      T num = static_cast<T>(1);
8      while (num != static_cast<T>(0)) {
9          num = static_cast<T>(num / 2);
10         k += 1;
11     }
12     std::cout << typeid(T).name() << " машинный ноль = 2^-" << k << std::endl;
13 }
14
15 template <typename T>
16 void print_infinity() {
17     int k = 0;
18     T num = static_cast<T>(1);
19     while (num < std::numeric_limits<T>::max()) {
20         num = static_cast<T>(num * 2);
21         k += 1;
22     }
23     std::cout << typeid(T).name() << " машинная бесконечность = 2^" << k << std::endl;
24 }
25
26 template <typename T>
27 void print_epsilon() {
28     int k = 0;
29     T num = static_cast<T>(1);
30     while (static_cast<T>(1.) + num > static_cast<T>(1.)) {
31         num = static_cast<T>(num / 2);
32         k += 1;
33     }
34     std::cout << typeid(T).name() << " машинное эpsilon = 2^-" << k << std::endl;
35 }
36
37 int main() {
38     print_zero<float>();
39     print_infinity<float>();
40     print_epsilon<float>();
41     std::cout << std::endl;
42
43     print_zero<double>();
44     print_infinity<double>();
45     print_epsilon<double>();
46     std::cout << std::endl;
47
48     print_zero<long double>();
```



```
49     print_infinity<long double>();  
50     print_epsilon<long double>();  
51     std::cout << std::endl;  
52  
53     return 0;  
54 }
```

3.4 Вывод кода на Python

```
<class 'numpy.float32'> машинный ноль = 2-150  
<class 'numpy.float32'> машинная бесконечность = 2128  
<class 'numpy.float32'> машинное эpsilon = 2-24  
  
<class 'numpy.float64'> машинный ноль = 2-1075  
<class 'numpy.float64'> машинная бесконечность = 21024  
<class 'numpy.float64'> машинное эpsilon = 2-53  
  
<class 'numpy.longdouble'> машинный ноль = 2-1075  
<class 'numpy.longdouble'> машинная бесконечность = 21024  
<class 'numpy.longdouble'> машинное эpsilon = 2-53
```

3.5 Вывод кода на C++

```
f машинный ноль = 2-150  
f машинная бесконечность = 2128  
f машинное эpsilon = 2-24  
  
d машинный ноль = 2-1075  
d машинная бесконечность = 21024  
d машинное эpsilon = 2-53  
  
e машинный ноль = 2-1075  
e машинная бесконечность = 21024  
e машинное эpsilon = 2-53
```

3.6 Сравнение результатов

Видно, что и машинный ноль, и машинное эpsilon, и машинная бесконечность совпадают для Python и для C++.