

Отчет по лабораторной работе №6

Дисциплина: Научное программирование

Выполнила Дяченко Злата Константиновна, НПМмд-02-22

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Шаг 1	7
3.2	Шаг 2	8
3.3	Шаг 3	9
3.4	Шаг 4	10
3.5	Шаг 5	10
4	Выводы	13

List of Figures

3.1	Задание функции и индексной переменной	7
3.2	Оценка функции	8
3.3	Вычисление членов ряда и частичных сумм	8
3.4	График членов ряда и частичных сумм	9
3.5	Сумма ряда	9
3.6	Вычисление интеграла	10
3.7	Скрипт для вычисления интеграла по правилу средней точки с помощью цикла	10
3.8	Результат выполнения скрипта	11
3.9	Скрипт для вычисления интеграла по правилу средней точки с помощью векторов	11
3.10	Результат выполнения скрипта	11
3.11	Время выполнения скрипта для каждого скрипта	12

List of Tables

1 Цель работы

Научиться работать с пределами, последовательностями и рядами, выполнять численное интегрирование с помощью Octave.

2 Задание

Оценить предел, вычислить частичную сумму и сумму ряда, вычислить интеграл с помощью встроенной функции и по правилу средней точки .

3 Выполнение лабораторной работы

3.1 Шаг 1

Рассмотрим предел $\lim_{n \rightarrow \infty} (1 + \frac{1}{n})^n$. Для его оценки определила функцию как анонимную функцию и создала индексную переменную, состоящую из целых чисел от 0 до 9. Соответствующие команды показаны на Рисунке 1 (рис - fig. 3.1).

```
>> f=@(n) (1+1./n).^n
f =

@(n) (1 + 1 ./ n) .^ n

>> k=[0:1:9]'
k =

0
1
2
3
4
5
6
7
8
9
```

Figure 3.1: Задание функции и индексной переменной

В качестве входных значений будем использовать степени 10. Получаем результат, показанный на Рисунке 2 (рис - fig. 3.2), то есть предел сходится к конечному значению 2,71828...

```

>> format long
>> n=10.^k
n =

         1
        10
       100
      1000
     10000
    100000
   1000000
  10000000
 100000000
1000000000

>> f(n)
ans =

 2.000000000000000
 2.593742460100002
 2.704813829421529
 2.716923932235520
 2.718145926824356
 2.718268237197528
 2.718280469156428
 2.718281693980372
 2.718281786395798
 2.718282030814509

>> format

```

Figure 3.2: Оценка функции

3.2 Шаг 2

Рассмотрим ряд $\sum_{n=2}^{\infty} a_n$, n -й член равен $a_n = \frac{1}{n(n+2)}$. Используя команды Octave, представленные на Рисунке 3 (рис - fig. 3.3), вычислила члены от 2 до 11, а затем, используя цикл, получила частичные суммы. Слагаемые и частичные суммы построила на графике, который показан на Рисунке 4 (рис - fig. 3.4).

```

>> n=[2:1:11]';
>> a=1./(n.*(n+2))
a =

 1.2500e-01
 6.6667e-02
 4.1667e-02
 2.8571e-02
 2.0833e-02
 1.5873e-02
 1.2500e-02
 1.0101e-02
 8.3333e-03
 6.9930e-03

>> for i =1:10
s(i)=sum(a(1:i));
end
>> s'
ans =

 0.1250
 0.1917
 0.2333
 0.2619
 0.2827
 0.2986
 0.3111
 0.3212
 0.3295
 0.3365

>> plot(n, a, 'o', n, s, '+')
>> grid on
>> legend ('terms', 'partial sums')

```

Figure 3.3: Вычисление членов ряда и частичных сумм

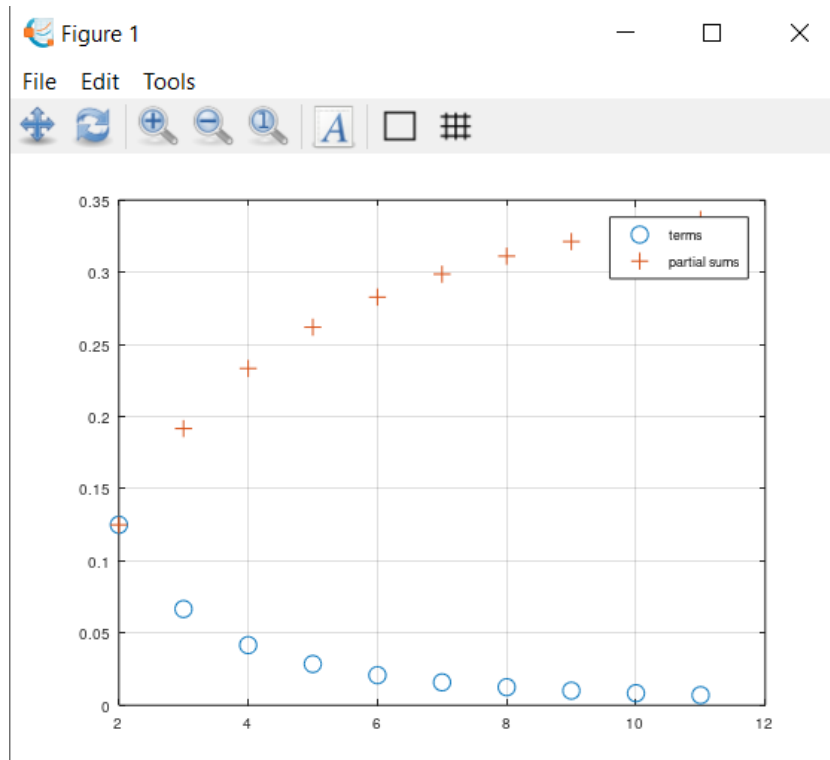


Figure 3.4: График членов ряда и частичных сумм

3.3 Шаг 3

Для нахождения суммы первых 1000 членов гармонического ряда $\sum_{n=1}^{1000} \frac{1}{n}$ сгенерировала члены ряда как вектор, а затем взяла их сумму, что показано на Рисунке 5 (рис - fig. 3.5). Сумма получилась равна 7,4855.

```
>> n=[1:1:1000];
>> a=1./n;
>> sum(a)
ans = 7.4855
>>
```

Figure 3.5: Сумма ряда

3.4 Шаг 4

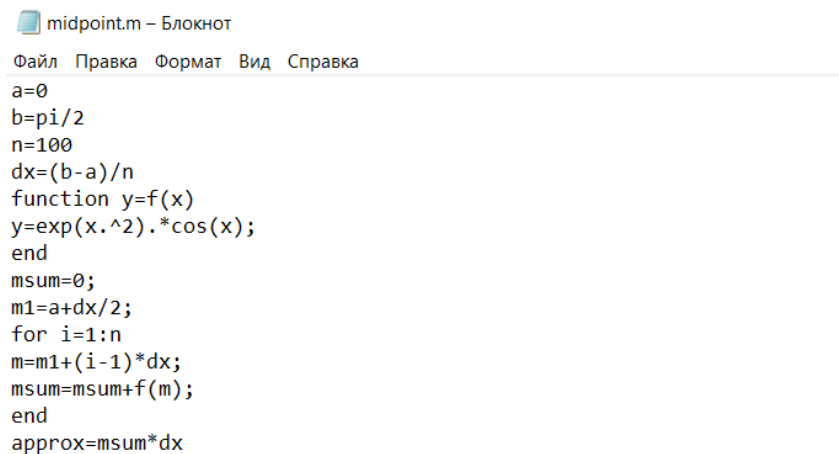
Для вычисления интеграла $\int_0^{\pi/2} e^{x^2} \cos(x) dx$ использовала встроенную команду, показанную на Рисунке 6 (рис - fig. 3.6).

```
>> function y=f(x)
y=exp(x.^2).*cos(x);
end
>> quad('f', 0, pi/2)
ans = 1.8757
>>
```

Figure 3.6: Вычисление интеграла

3.5 Шаг 5

Для вычисления того же интеграла использовала также правило средней точки для $n=100$. Для этого написала скрипт, использующий цикл, который показан на Рисунке 7 (рис - fig. 3.7). Результат выполнения данного скрипта показан на Рисунке 8 (рис - fig. 3.8). показано увеличение графа в два раза.



```
midpoint.m – Блокнот
Файл Правка Формат Вид Справка
a=0
b=pi/2
n=100
dx=(b-a)/n
function y=f(x)
y=exp(x.^2).*cos(x);
end
msum=0;
m1=a+dx/2;
for i=1:n
m=m1+(i-1)*dx;
msum=msum+f(m);
end
approx=msum*dx
```

Figure 3.7: Скрипт для вычисления интеграла по правилу средней точки с помощью цикла

```
>> midpoint
a = 0
b = 1.5708
n = 100
dx = 0.015708
approx = 1.8758
>> |
```

Figure 3.8: Результат выполнения скрипта

Так как Octave является векторным языком, написала скрипт, вычисляющий аппроксимацию средней точки с использованием векторов. Скрипт представлен на Рисунке 9 (рис - fig. 3.9), а результат его выполнения на Рисунке 10 (рис - fig. 3.10).

```
*midpoint_v.m – Блокнот
Файл  Правка  Формат  Вид  Справка
a=0
b=pi/2
n=100
dx=(b-a)/n
function y=f(x)
y=exp(x.^2).*cos(x);
end
m=[a+dx/2:dx:b-dx/2];
M=f(m);
approx=dx*sum(M)
```

Figure 3.9: Скрипт для вычисления интеграла по правилу средней точки с помощью векторов

```
>> midpoint_v
a = 0
b = 1.5708
n = 100
dx = 0.015708
approx = 1.8758
```

Figure 3.10: Результат выполнения скрипта

Полученные результаты отличаются в десятитысячной доле. Я также сравнила

время выполнения каждого скрипта, что показано на Рисунке 11 (рис - fig. 3.11). Вычисление с помощью векторов работает чуть больше чем в три раза быстрее вычисления с использованием цикла.

```
>> tic; midpoint; toc
a = 0
b = 1.5708
n = 100
dx = 0.015708
approx = 1.8758
Elapsed time is 0.00568008 seconds.
>> tic; midpoint_v; toc
a = 0
b = 1.5708
n = 100
dx = 0.015708
approx = 1.8758
Elapsed time is 0.00181198 seconds.
```

Figure 3.11: Время выполнения скрипта для каждого скрипта

4 Выводы

Я научилась работать с пределами, последовательностями и рядами, выполнять численное интегрирование с помощью Octave. Результаты работы находятся в репозитории на GitHub, а также есть скринкаст выполнения лабораторной работы.