

Лабораторная работа №6

Ишанова А.И. группа НФИ-02-19

Содержание

1	Цель работы	4
2	Задание работы	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
4.1	Пределы, последовательности и ряды	8
4.1.1	Предел	8
4.1.2	Частичные суммы	12
4.1.3	Сумма ряда	16
4.2	Численное интегрирование	18
4.2.1	Вычисление интегралов	18
4.2.2	Аппроксимирование суммами	19
5	Вывод	23
6	Библиография	24

List of Figures

4.1	подготовка к лабораторной работе	8
4.2	начало журналирования	8
4.3	определение функции	8
4.4	индексная переменная	9
4.5	определение входных значений	10
4.6	оценивание функции	11
4.7	определение значений индексов	13
4.8	вычисление членов ряда	14
4.9	вычисление частичных сумм	15
4.10	команды построения графика	16
4.11	график значений членов ряда и частичной суммы от 2 до них . .	16
4.12	определение значений индексов (часть)	17
4.13	определение значений членов ряда (часть)	17
4.14	определение значений индексов	17
4.15	задание функции и расчет определенного интеграла от 0 до $\pi/2$	18
4.16	задание анонимной функции и расчет определенного интеграла	18
4.17	создание файла midpoint.m	19
4.18	содержимое файла midpoint.m	19
4.19	работа midpoint.m	20
4.20	создание файла midpoint_v.m	20
4.21	содержимое файла midpoint_v.m	21
4.22	работа midpoint_v.m	21
4.23	сравнение midpoint.m и midpoint_v.m по времени	22

1 Цель работы

Научиться работать в Octave с пределами, последовательностями, рядами и численным интегрированием.

2 Задание работы

Выполнить лабораторную работу и сделать отчет по лабораторной работе в форматах md, docx и pdf.

3 Теоретическое введение

В octave можно реализовать пределы, последовательности и ряды, а так же считать определенный интеграл.

- Предел — одно из основных понятий математического анализа, на него опираются такие фундаментальные разделы анализа, как непрерывность, производная, интеграл, бесконечные ряды и др. Различают предел последовательности и предел функции.
- Последовательность — это пронумерованный набор каких-либо объектов, среди которых допускаются повторения, причём порядок объектов имеет значение. Нумерация чаще всего происходит натуральными числами.
- Ряд, называемый также бесконечная сумма — одно из центральных понятий математического анализа. В простейшем случае ряд записывается как бесконечная сумма чисел:

$a_1 + a_2 + a_3 + \dots + a_n + \dots$; краткая запись $\sum_{n=1}^{\infty} a_n$ (иногда нумерацию слагаемых начинают не с 1, а с 0)

- Определённый интеграл — одно из основных понятий математического анализа, один из видов интеграла. Определённый интеграл является числом, равным пределу сумм особого вида (интегральных сумм). Геометрически определённый интеграл выражает площадь «криволинейной трапеции», ограниченной графиком функции.

- Квадратура (лат. quadratura, придание квадратной формы) — математический термин, первоначально обозначавший нахождение площади какой-либо фигуры или поверхности. То есть, мы разбиваем/приближаем площадь под графиком к набору прямоугольников одинаковой ширины.
- правило средней точки:

Аппроксимация f в середине интервалов дает $f(a + \Delta x / 2)$ для первого интервала, для следующий $f(a + 3\Delta x / 2)$, и так далее до $f(b - \Delta x / 2)$. Суммирование площадей дает: $A_{mid} = \Delta x [f(a + \frac{\Delta x}{2}) + f(a + \frac{3\Delta x}{2}) + \dots + f(b - \frac{\Delta x}{2})]$

4 Выполнение лабораторной работы

1. Создаем каталог для работы в папке laboratory. (mkdir) (fig. 4.1)

```
(base) alinaishanova@iMac-Alina ~ % cd work  
(base) alinaishanova@iMac-Alina work % cd 2020-2021/"Введение в научное программ  
ирование"/laboratory  
(base) alinaishanova@iMac-Alina laboratory % mkdir lab06
```

Figure 4.1: подготовка к лабораторной работе

2. Начинаем сессию журналирования. (fig. 4.2)

```
>> diary on
```

Figure 4.2: начало журналирования

4.1 Пределы, последовательности и ряды

4.1.1 Предел

1. Для расчета предела функции, нужно сначала определить функцию. (fig. 4.3)

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

Figure 4.3: определение функции

2. Создаем индексную переменную. (fig. 4.4)

```
>> k = [0:1:9]'  
k =  
  
    0  
    1  
    2  
    3  
    4  
    5  
    6  
    7  
    8  
    9
```

Figure 4.4: индексная переменная

3. Берем степени 10, которые будут входными значениями. (fig. 4.5)

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

```
1  
10  
100  
1000  
10000  
100000  
1000000  
10000000  
100000000  
1000000000  
10000000000
```

Figure 4.5: определение входных значений

4. Оцениваем $f(n)$. (fig. 4.6)

```
>> f(n)
ans =

2.0000000000000000
2.593742460100002
2.704813829421528
2.716923932235594
2.718145926824926
2.718268237192297
2.718280469095753
2.718281694132082
2.718281798347358
2.718282052011560

>> format
```

Figure 4.6: оценивание функции

4.1.2 Частичные суммы

1. Задаем вектор входных значений/индексов. (fig. 4.7)

```
>> n = [2:1:11] '  
n =  
  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11
```

Figure 4.7: определение значений индексов

2. Вычисляем члены ряда. (fig. 4.8)

```
>> 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  
  
format long
```

Figure 4.8: вычисление членов ряда

3. Пишем цикл для суммирования членов ряда, получаем суммы 2 член, 2-3 член, 3-4 член и тд. до 2-11 член. (fig. 4.9)

```
>> 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
```

Figure 4.9: вычисление частичных сумм

4. Строим график с полученными данными. (fig. 4.10 и fig. 4.11)

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

Figure 4.10: команды построения графика

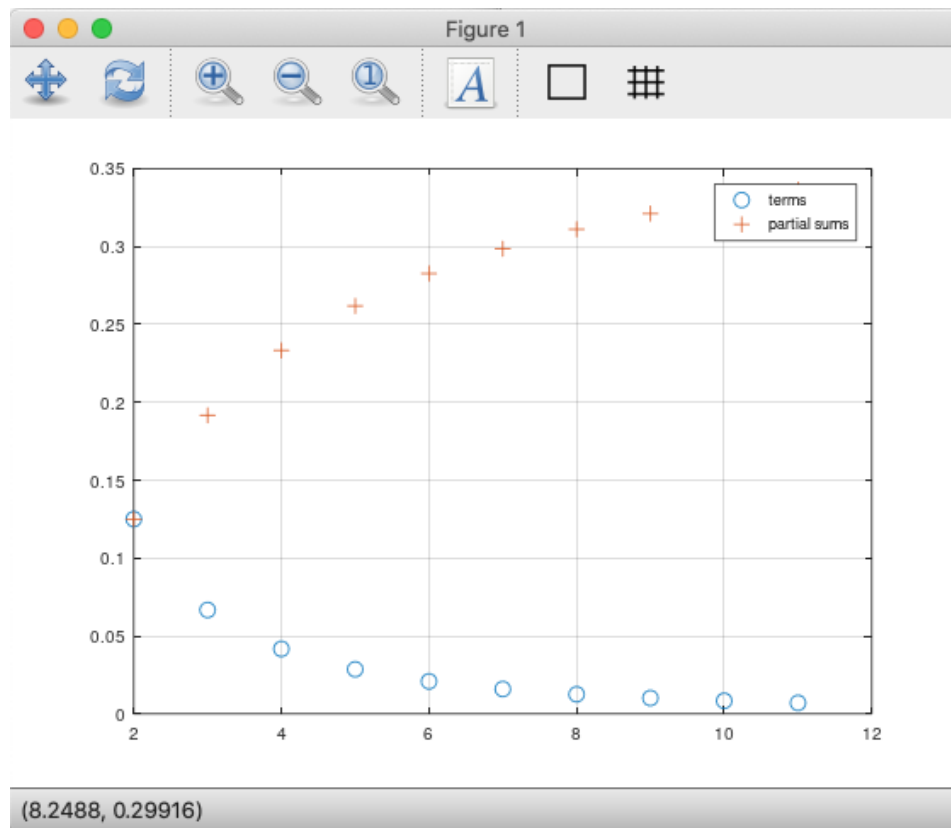


Figure 4.11: график значений членов ряда и частичной суммы от 2 до них

4.1.3 Сумма ряда

1. Задаем вектор входных значений/индексов. (fig. 4.12)


```
>> n = [1:1:1000]
n =
```

Columns 1 through 10:

1	2	3	4
---	---	---	---

Figure 4.12: определение значений индексов (часть)

2. Задаем гармонический ряд. Задаем вектор входных значений/индексов.
(fig. 4.13)

```
>> a = 1 ./ n
a =
Columns 1 through 5:
    1.0000e+00    5.0000e-01    3.3333e-01    2.5000e-01    2.0000e-01
Columns 6 through 10:
    1.6667e-01    1.4286e-01    1.2500e-01    1.1111e-01    1.0000e-01
```

Figure 4.13: определение значений членов ряда (часть)

3. Вычисляем сумму ряда. Задаем вектор входных значений/индексов.
(fig. 4.14)

```
>> sum(a)
ans = 7.4855
```

Figure 4.14: определение значений индексов

4.2 Численное интегрирование

4.2.1 Вычисление интегралов

1. Определяем функцию, чей интеграл мы будем считать, и считаем определенный интеграл командой `quad`. (fig. 4.15)

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

Figure 4.15: задание функции и расчет определенного интеграла от 0 до $\pi/2$

2. Рассчитаем определенный интеграл, используя анонимную функцию. (fig. 4.16)

```
>> f = @(x) exp(x.^2) .* cos(x)
f =

@(x) exp (x.^2) .* cos (x)

>> quad('f', 0, pi/2)
ans = 1.8757
>> quad(f, 0, pi/2)
ans = 1.8757
```

Figure 4.16: задание анонимной функции и расчет определенного интеграла

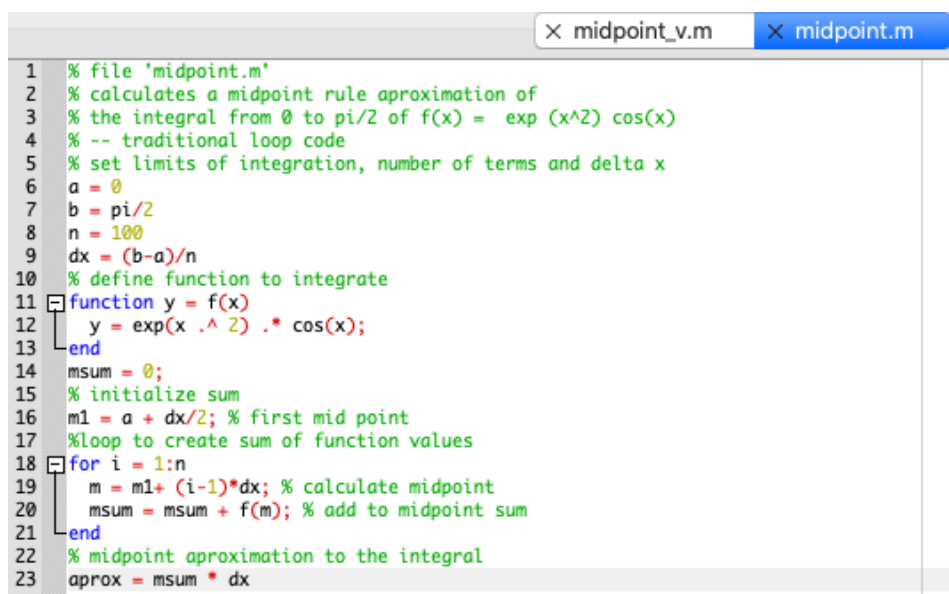
4.2.2 Аппроксимирование суммами

1. Создаем файл Octave. (fig. 4.17)

```
(base) alinaishanova@iMac-Alina laboratory % touch midpoint.m
```

Figure 4.17: создание файла midpoint.m

2. В нем записываем реализацию метода средней точки с помощью цикла. (fig. 4.18)



```
1 % file 'midpoint.m'
2 % calculates a midpoint rule approximation of
3 % the integral from 0 to pi/2 of f(x) = exp(x^2) cos(x)
4 % -- traditional loop code
5 % set limits of integration, number of terms and delta x
6 a = 0
7 b = pi/2
8 n = 100
9 dx = (b-a)/n
10 % define function to integrate
11 function y = f(x)
12 y = exp(x.^2) .* cos(x);
13 end
14 msum = 0;
15 % initialize sum
16 m1 = a + dx/2; % first mid point
17 %loop to create sum of function values
18 for i = 1:n
19 m = m1 + (i-1)*dx; % calculate midpoint
20 msum = msum + f(m); % add to midpoint sum
21 end
22 % midpoint approximation to the integral
23 aprox = msum * dx
```

Figure 4.18: содержимое файла midpoint.m

3. Запускаем файл. (fig. 4.19)

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

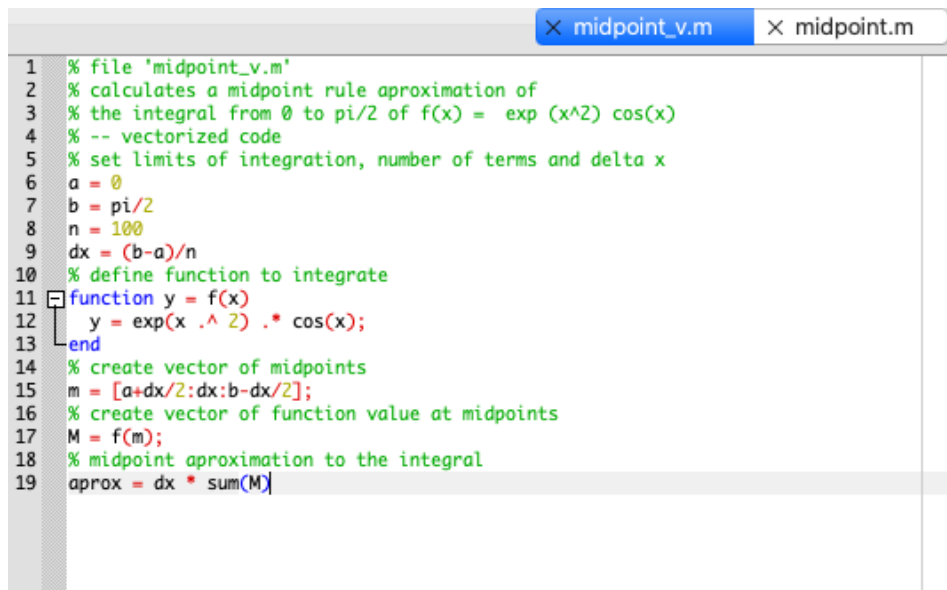
Figure 4.19: работа midpoint.m

4. Создаем второй файл Octave. (fig. 4.20)

```
(base) alinaishanova@iMac-Alina lab06 % touch midpoint_v.m
```

Figure 4.20: создание файла midpoint_v.m

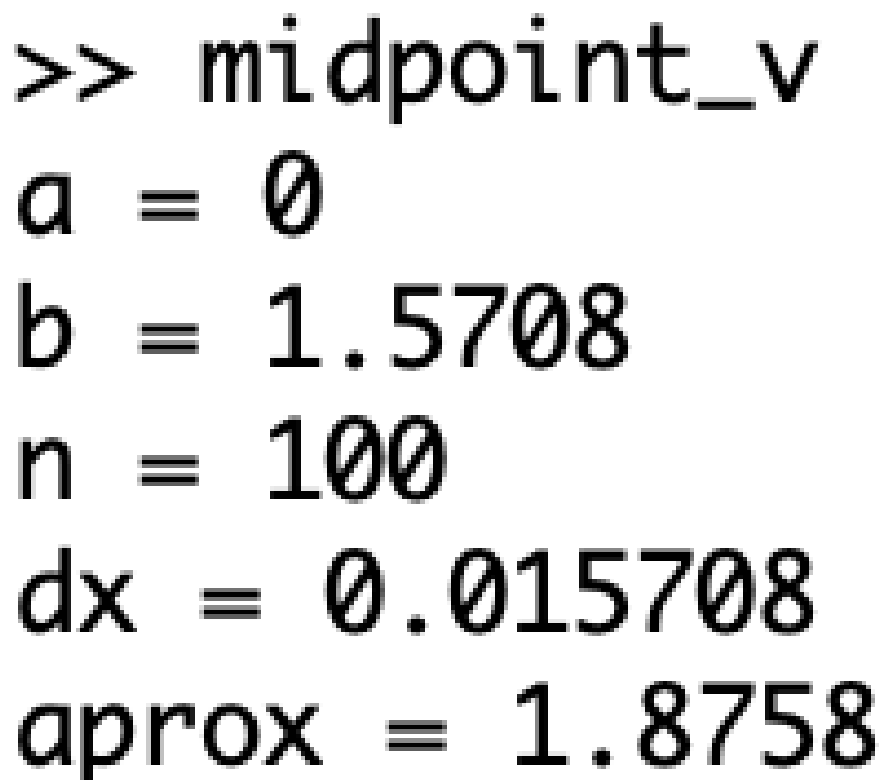
5. В нем записываем векторизованную реализацию метода средней точки.
(fig. 4.21)



```
1 % file 'midpoint_v.m'
2 % calculates a midpoint rule aproximation of
3 % the integral from 0 to pi/2 of f(x) = exp (x^2) cos(x)
4 % -- vectorized code
5 % set limits of integration, number of terms and delta x
6 a = 0
7 b = pi/2
8 n = 100
9 dx = (b-a)/n
10 % define function to integrate
11 function y = f(x)
12     y = exp(x.^2) .* cos(x);
13 end
14 % create vector of midpoints
15 m = [a+dx/2:dx:b-dx/2];
16 % create vector of function value at midpoints
17 M = f(m);
18 % midpoint aproximation to the integral
19 aprox = dx * sum(M)
```

Figure 4.21: содержимое файла midpoint_v.m

6. Запускаем файл. (fig. 4.22)



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

Figure 4.22: работа midpoint_v.m

7. Для того, чтобы увидеть разницу между двумя этими подходами, засекаем время выполнения файлов. (fig. 4.23)

```
>> tic; midpoint; toc
a = 0
b = 1.5708
n = 100
dx = 0.015708
aprox = 1.8758
Elapsed time is 0.00371599 seconds.
>> tic; midpoint_v; toc
a = 0
b = 1.5708
n = 100
dx = 0.015708
aprox = 1.8758
Elapsed time is 0.000440836 seconds.
```

Figure 4.23: сравнение midpoint.m и midpoint_v.m по времени

Видим, что векторизированный код работает на порядок быстрее.

5 Вывод

В ходе выполнения работы мы научились считать пределы и частичные суммы рядов, считать определенный интеграл встроенной окмандой `quad` и методом средней точки, а так же увидели разницу в скорости работы трандиционного кода (с циклами) и векторизированного кода.

6 Библиография

- [illegible]