

**Отчёт по лабораторной работе №6.
Пределы, последовательности и ряды,
интегралы.**

Предмет: научное программирование

Александр Сергеевич Баклашов

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
3.1	Пределы, последовательности и ряды	7
3.1.1	Частичные суммы	10
3.1.2	Сумма ряда	11
3.2	Численное интегрирование	12
3.2.1	Вычисление интегралов	12
3.2.2	Аппроксимирование суммами	13
4	Вывод	19
5	Библиография	20

Список иллюстраций

3.1	Пределы, последовательности и ряды	9
3.2	Частичные суммы	11
3.3	Сумма ряда	12
3.4	Вычисление интегралов	13
3.5	midpoint.m	14
3.6	midpoint	15
3.7	midpoint_v.m	16
3.8	midpoint_v	17
3.9	Сравнение	18

Список таблиц

1 Цель работы

Изучить пределы, последовательности и ряды и интегралы в Octave.

2 Теоретическое введение

GNU Octave — свободная программная система для математических вычислений, использующая совместимый с MATLAB язык высокого уровня.

Предоставляет интерактивный командный интерфейс для решения линейных и нелинейных математических задач, а также проведения других численных экспериментов. Кроме того, Octave можно использовать для пакетной обработки. Язык Octave оперирует арифметикой вещественных и комплексных скаляров, векторов и матриц, имеет расширения для решения линейных алгебраических задач, нахождения корней систем нелинейных алгебраических уравнений, работы с полиномами, решения различных дифференциальных уравнений, интегрирования систем дифференциальных и дифференциально-алгебраических уравнений первого порядка, интегрирования функций на конечных и бесконечных интервалах. Этот список можно легко расширить, используя язык Octave (или используя динамически загружаемые модули, созданные на Си, C++, Фортране и других). [1]

3 Выполнение лабораторной работы

3.1 Пределы, последовательности и ряды

Octave - полноценный язык программирования, поддерживающий множество типов циклов и условных операторов. Однако, поскольку это векторный язык, многие вещи, которые можно было бы сделать с помощью циклов, можно векторизовать. Под векторизованным кодом мы понимаем следующее: вместо того, чтобы писать цикл для многократной оценки функции, мы сгенерируем вектор входных значений, а затем оценим функцию с использованием векторного ввода. В результате получается код, который легче читать и понимать, и он выполняется быстрее благодаря эффективным алгоритмам для матричных операций.

Рассмотрим предел:

$$\lim_{n \rightarrow \infty} = \left(1 + \frac{1}{n}\right)^n$$

Оценим это выражение. Нужно определить функцию. Есть несколько способов сделать это. Метод, который мы здесь используем, называется анонимной функцией. Это хороший способ быстро определить простую функцию.

1. Определим анонимную функцию (Обратите внимание на использование поэлементных операций. Мы назвали функцию `f`. Входная переменная обозначается знаком `@`, за которым следует переменная в скобках. Следующее выражение будет использоваться при оценке функции. Теперь `f` можно использовать как любую функцию в Octave.)

2. Далее мы создаём индексную переменную, состоящую из целых чисел от 0 до 9
3. Синтаксис [0:1:9] создает вектор строки, который начинается с 0 и увеличивается с шагом от 1 до 9. Обратите внимание, что мы использовали операцию транспонирования просто потому, что наши результаты будут легче читать как векторы-столбцы. Теперь мы возьмём степени 10, которые будут входными значениями, а затем оценим $f(n)$.

Все пункты показаны на рисунке (рис. 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

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

         1
         10
        100
       1000
      10000
     100000
    1000000
   10000000
  100000000
 1000000000

>> f (n)
ans =

 2.0000000000000000
 2.593742460100002
 2.704813829421529
 2.716923932235520
 2.718145926824356
 2.718268237197528
 2.718280469156428
 2.718281693980372
 2.718281786395798
 2.718282030814509

>> format

```

Рис. 3.1: Пределы, последовательности и ряды

Предел сходится к конечному значению, которое составляет приблизительно 2,71828... Подобные методы могут быть использованы для численного исследования последовательностей и рядов.

3.1.1 Частичные суммы

1. Пусть $a \sum_{n=2}^{\infty} a_n$ — ряд, n -й член равен

$$a_n = \frac{1}{n(n+2)}$$

- . Для этого мы определим индексный вектор n от 2 до 11, а затем вычислим члены.
2. Если мы хотим знать частичную сумму, нам нужно только написать `sum(a)`. Если мы хотим получить последовательность частичных сумм, нам нужно использовать цикл. Мы будем использовать цикл `for` с индексом i от 1 до 10. Для каждого i мы получим частичную сумму последовательности a_n от первого слагаемого до i -го слагаемого. На выходе получается 10-элементный вектор этих частичных сумм.
3. Наконец, мы построим слагаемые и частичные суммы для $2 \leq n \leq 11$.

Все пункты показаны на рисунке(рис. 3.2)

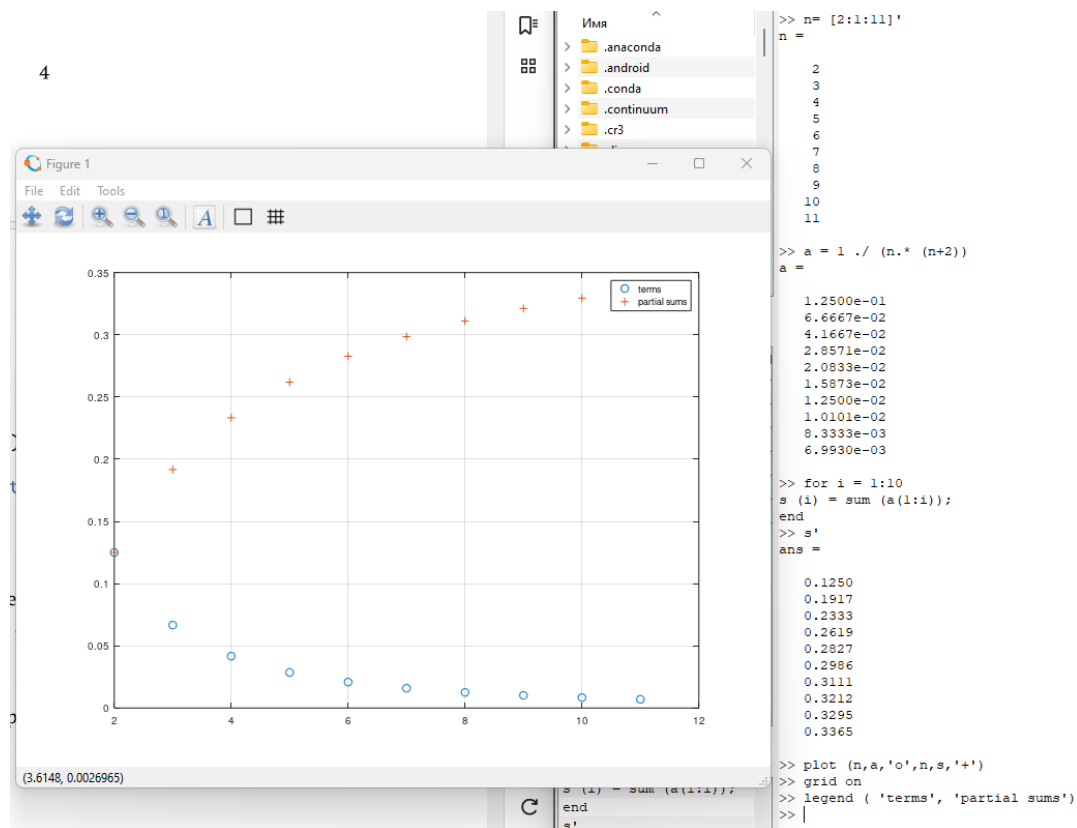


Рис. 3.2: Частичные суммы

3.1.2 Сумма ряда

1. Найдём сумму первых 1000 членов гармонического ряда:

$$\sum_{n=1}^{1000} \frac{1}{n}$$

Нам нужно только сгенерировать члены как ряда вектор, а затем взять их сумму. (рис. 3.3)

```

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

```

Рис. 3.3: Сумма ряда

3.2 Численное интегрирование

3.2.1 Вычисление интегралов

Octave имеет несколько встроенных функций для вычисления определённых интегралов. Мы будем использовать команду `quad` (сокращение от слова квадратура).

1. Вычислим интеграл:

$$\int_0^{\pi/2} e^{x^2} \cos(x) dx$$

Синтаксис команды - `quad('f', a, b)`. Нам нужно сначала определить функцию.

Обратите внимание, что функция `exp(x)` используется для e^x . Мы использовали конструкцию `function ... end`. Мы могли бы также использовать анонимную

функцию (сделайте это). Обратите внимание, что кавычки вокруг имени f не используются, если используется анонимная функция. (рис. 3.4)

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

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

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

Рис. 3.4: Вычисление интегралов

3.2.2 Аппроксимирование суммами

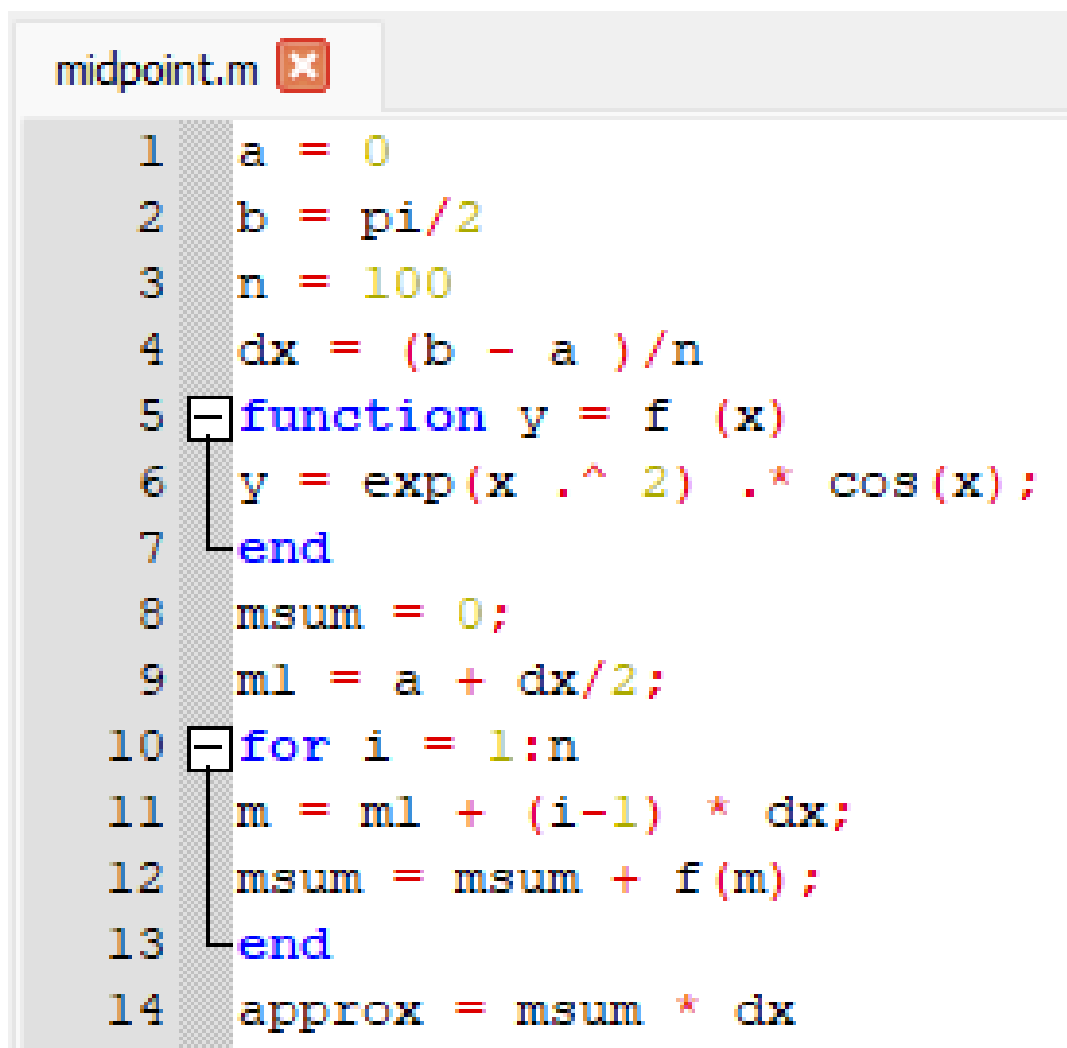
1. Правило средней точки, правило трапеции и правило Симпсона являются общими алгоритмами, используемыми для численного интегрирования. Напишем скрипт, чтобы вычислить интеграл

$$\int_0^{\pi/2} e^{x^2} \cos(x) dx$$

по правилу средней точки для $n = 100$.

Стратегия заключается в использовании цикла, который добавляет значение функции к промежуточной сумме с каждой итерацией. В конце сумма умножается на Δx .

Введём код в текстовом файле и назовём его midpoint.m. (рис. 3.5)



```
midpoint.m
1  a = 0
2  b = pi/2
3  n = 100
4  dx = (b - a) / n
5  function y = f(x)
6      y = exp(x.^2) .* cos(x);
7  end
8  msum = 0;
9  m1 = a + dx/2;
10 for i = 1:n
11     m = m1 + (i-1) * dx;
12     msum = msum + f(m);
13 end
14 approx = msum * dx
```

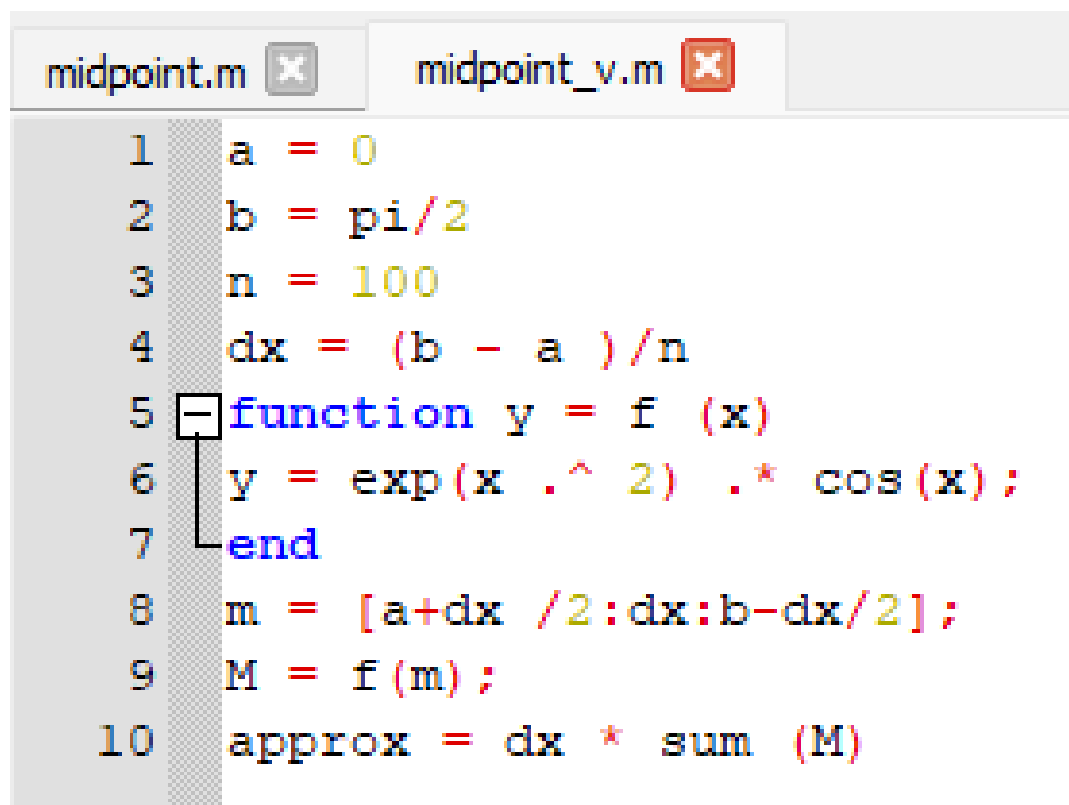
Рис. 3.5: midpoint.m

Он должен быть помещён в ваш рабочий каталог, а затем его можно запустить, набрав `midpoint` в командной строке. (рис. 3.6)

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

Рис. 3.6: midpoint

2. Традиционный код работает хорошо, но поскольку Octave является векторным языком, также можно писать векторизованный код, который не требует каких-либо циклов. Создадим вектор x-координат средних точек. Затем мы оцениваем f по этому вектору средней точки, чтобы получить вектор значений функции. Аппроксимация средней точки - это сумма компонент вектора, умноженная на Δx . (рис. 3.7)



```
midpoint.m x midpoint_v.m x
1 a = 0
2 b = pi/2
3 n = 100
4 dx = (b - a) / n
5 function y = f(x)
6 y = exp(x.^2) .* cos(x);
7 end
8 m = [a+dx/2:dx:b-dx/2];
9 M = f(m);
10 approx = dx * sum(M)
```

Рис. 3.7: midpoint_v.m

Запустим его. (рис. 3.8)


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

Рис. 3.8: midpoint_v

3. Сравним время выполнения для каждой реализации.(рис. 3.9)

```
>> tic; midpoint; toc
a = 0
b = 1.5708
n = 100
dx = 0.015708
approx = 1.8758
Elapsed time is 0.00368595 seconds.
>> tic; midpoint_v; toc
a = 0
b = 1.5708
n = 100
dx = 0.015708
approx = 1.8758
Elapsed time is 0.00247002 seconds.
>> |
```

Рис. 3.9: Сравнение

4 Вывод

В ходе данной лабораторной работы я изучил пределы, последовательности и ряды и интегралы в Octave.

5 Библиография

1. Лабораторная работа №6. - 10 с. [Электронный ресурс]. М. URL: Лабораторная работа №6. (Дата обращения: 22.11.2023).