РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

***дисциплина*: Научное программирование**

Лабораторная работа №6

Выполнил: Маслов Александр

Группа: НФИмд-02-20

С/б: 1032202156

Москва

2020

Цель работы:

Рассмотреть с помощью Octave пределы, последовательности и ряды, а также два метода оценки функции: с помощью циклов и с помощью вектора входных значений и сравнить их.

Ход работы:

Пределы, последовательности и ряды

Рассмотрим предел:

$$\lim_{n \to \infty} (1 + \frac{1}{n})^n$$

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.

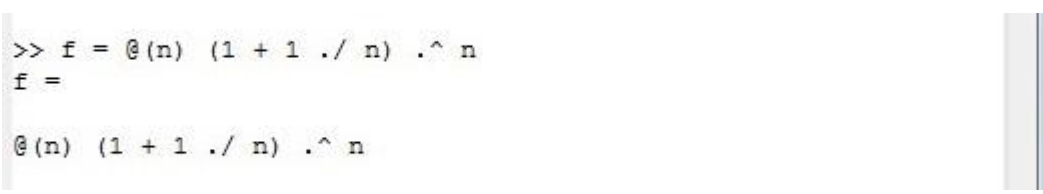

Рисунок 1. Определяем функцию f

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. Далее мы создаём индексную переменную, состоящую из целых чисел от 0 до 9:

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

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

Рисунок 2. Индексная переменная k

Синтаксис [0:1:9] создает вектор строки, который начинается с 0 и увеличивается с шагом от 1 до 9. Использовали операцию транспонирования. Теперь мы возьмём степени 10, которые будут входными значениями, а затем оценим f(n).

```
>> 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
```

Рисунок 3. Степени 10

Предел сходится к конечному значению, которое составляет при¬ близительно 2,71828

Частичные суммы

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.

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

   0.1250000
   0.0666667
   0.0416667
   0.0285714
   0.0208333
   0.0158730
   0.0125000
   0.0101010
   0.0083333
   0.0069930
```

Рисунок 4. Индексный вектор n и члены a

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.

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

   0.12500
   0.19167
   0.23333
   0.26190
   0.28274
   0.29861
   0.31111
   0.32121
   0.32955
   0.33654
```

Рисунок 5. Вектор частичных сумм

Наконец, мы построим слагаемые и частичные суммы для 2 < n < 11.

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

Рисунок 6. Построение графика

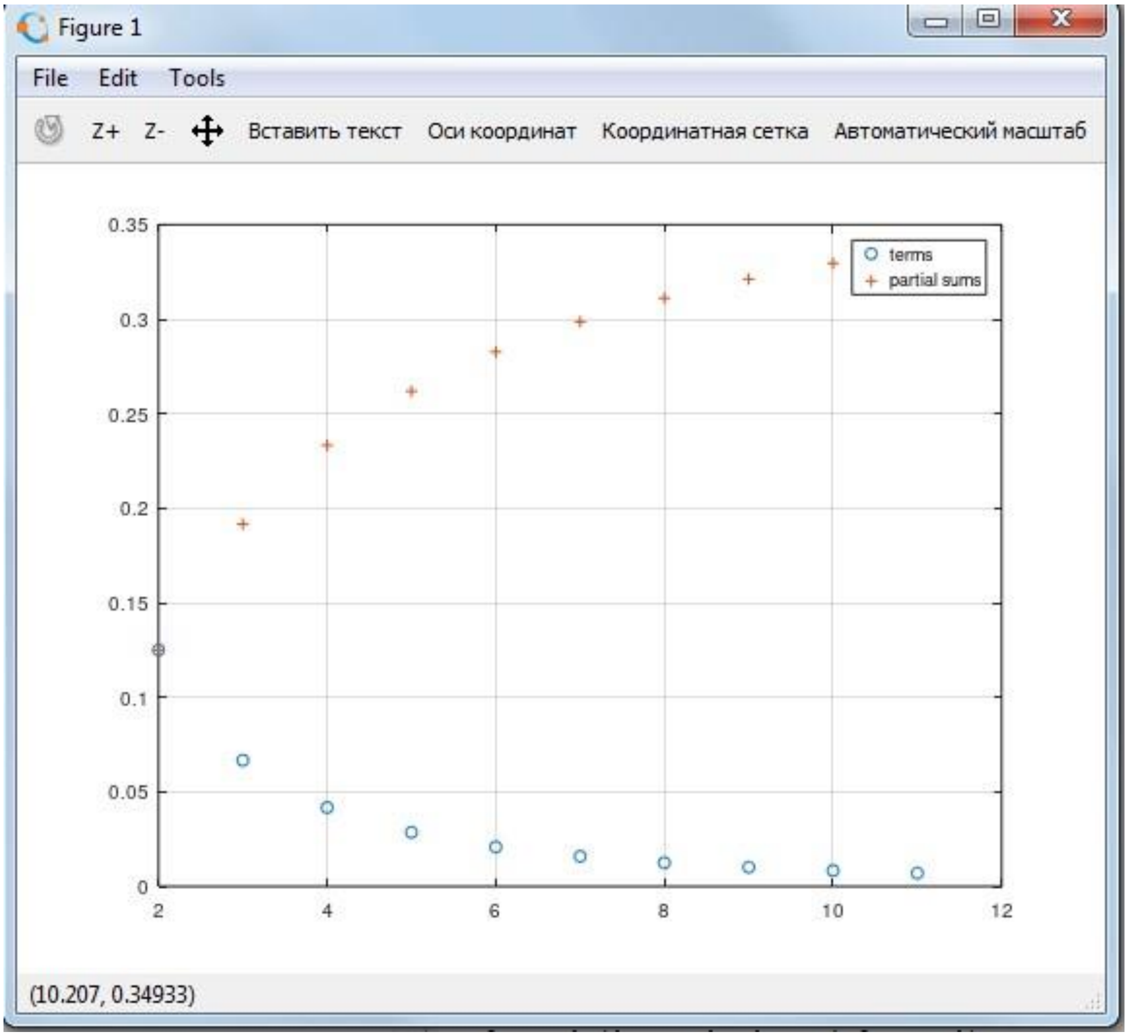

Рисунок 7. График

Сумма ряда

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.

```
>> n = [1:1:1000];
>> a = 1 ./ n;
>> um (a)
error: 'um' undefined near line 1 column 1
>> sum (a)
ans =  7.4855
```

Рисунок 8. Сумма ряда

Численное интегрирование

Вычисление интегралов

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

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

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

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

Рисунок 9. Вычисление интеграла

Обратите внимание, что функция exp (x) используется для e^{x}.Мы использовали конструкцию function . . . end.

Аппроксимирование суммами

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.

```
midpoint.m    midpoint_v.m

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
```

Рисунок 10. Код файла mibpoint.m

Ондолжен бытьпомещён вваш рабочий каталог,азатем его можно запустить,набрав midpoint в командной строке.

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

Рисунок 11. Запуск файла mibpoint.m

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.

```
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)
```

Рисунок 12. Код файла mibpoint_v.m

Запустим его:

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

Рисунок 13. Запуск файла mibpoint_v.m

Сравним результаты и время выполнения для каждой реализации.

```
>> tic; midpoint; toc
a = 0
b =  1.5708
n =  100
dx =  0.015708
approx =  1.8758
Elapsed time is 0.0159998 seconds.
>> tic; midpoint_v; toc
a = 0
b =  1.5708
n =  100
dx =  0.015708
approx =  1.8758
Elapsed time is 0.0079999 seconds.
>>
```

Рисунок 13. Результаты и время выполнения

Результаты обоих методов получились одинаковыми, но метод оценки функции с помощью цикла выполняется намного медленнее.

Вывод:

В процессе выполнения лабораторной работы с помощью Octave мы рассмотрели пределы, последовательности и ряды, а также два метода оценки функции: с помощью циклов и с помощью вектора входных значений и сравнили их.