

Лабораторная работа №6

Научное программирование

Дарижапов Тимур Андреевич

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	13
5	Список литературы	14

List of Tables

List of Figures

3.1	Вывод данных	7
3.2	Программа	8
3.3	Программа	9
3.4	программа	9
3.5	График	10
3.6	Код	10
3.7	Программа	10
3.8	Вариант 1	11
3.9	Вариант 2	11
3.10	Выводы	12

1 Цель работы

Изучение языка Octave, знакомство с методами работы с последовательностями, пределами, рядами.

2 Задание

Разобраться со спецификой языка и выполнить операции.

1. Пределы
2. Частичные суммы
3. Суммы ряда
4. Вычисление интегралов
5. Аппроксимирование суммами

3 Выполнение лабораторной работы

Для начала работы с программой включим журналирование сессии командой `diary on`. Затем приступим к выполнению первого этапа - работе с пределами. Определим анонимную функцию, создадим индексную переменную от 0 до 9 (рис. 3.1)

```
>> diary on
>> 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

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

     0
     1
     2
     3
     4
     5
     6
     7
     8
     9

>> format long
```

Figure 3.1: Вывод данных

Возьмём степени 10, которые будут входными значениями и оценим нашу

функцию. Результат: предел сходится к значению 2.718 (рис. 3.2)

```
>> n = 10 .^ k̄
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: Программа

Теперь определим частичные суммы ряда.

Для начала определим индексный вектор, а затем вычислим члены. Чтобы узнать частичные суммы, остаётся только дописать команду `sum`. Напишем это в цикле (рис. 3.3)


```

>> 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

```

Figure 3.3: Программа

Нарисуем получившееся (рис. 3.4)

```

>> plot (n,a,'o',n,s,'+')
Fontconfig error: Cannot load default config file:
\fonts.conf
< >> grind on
error: 'grind' undefined near line 1, column 1
>> grid on
>> legend ('terms', 'partical sums')

```

Figure 3.4: программа

Получили такой рисунок (рис. 3.5)

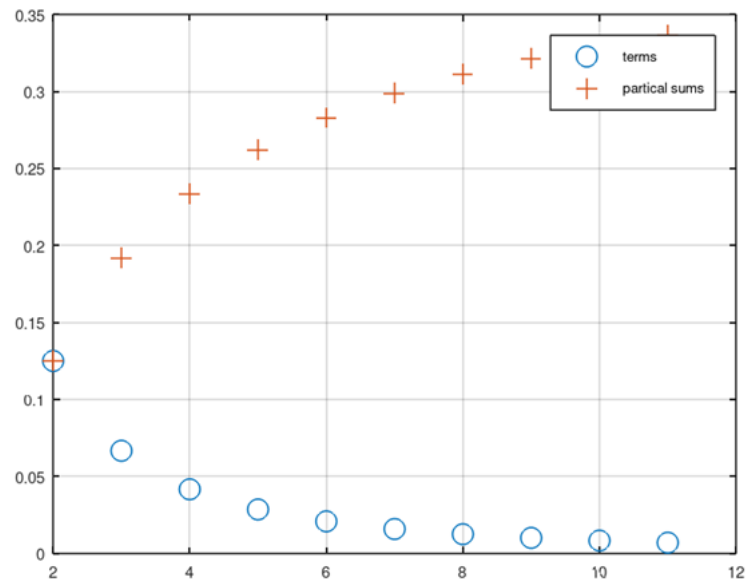


Figure 3.5: График

Теперь переходим к суммам ряда. Это сделать проще всего. Вычислим сумму первых 1000 членов гармонического ряда. Определим эти члены и посчитаем сумму (рис. 3.6)

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

Figure 3.6: Код

Переходим к разделу интегрирования. Для начала вычислим интеграл при помощи функции `quad`. Определим функцию и применим её (рис. 3.7)

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

Figure 3.7: Программа

И последний раздел: аппроксимирование суммами. Сделаем это двумя способами: циклами и с помощью векторов. Для этого напишем два варианта кода (рис. 3.8) (рис. 3.9)

```
a = 0
b = pi/2
n = 100
dx = (b-a)/n
% define function to integrate
function y = f (x)
y = exp(x.^2) .* cos(x);
end
msum = 0;
% initialize sum
m1 = a + dx/2; % first midpoint
% loop to create sum of function values
for i = 1:n
m = m1 + (i-1) * dx; % calculate midpoint
msum = msum + f (m); % add to midpoint sum
end
% midpoint approximation to the integral
approx = msum * dx
```

Figure 3.8: Вариант 1

```
% file 'midpoint_v.m'
% calculates a midpoint rule approximation of
% the integral from 0 to pi/2 of f(x) = exp(x^2) cos(x)
% -- vectorized code
% set limits of integration, number of terms and delta x
a = 0
b = pi/2
n = 100
dx = (b-a)/n
% define function to integrate
function y = f (x)
y = exp(x.^2) .* cos(x);
end
% create vector of midpoints
m = [a+dx/2:dx:b-dx/2];
% create vector of function values at midpoints
M = f(m);
% midpoint approximation to the integral
approx = dx * sum (M)
```

Figure 3.9: Вариант 2

Запустим их и сверим результаты (рис. 3.10)

```
>> tic; midpoint; toc
a = 0
b = 1.5708
n = 100
dx = 0.015708
approx = 1.8758
Elapsed time is 0.00229788 seconds.

>> tic; midpoint_v; toc
a = 0
b = 1.5708
n = 100
dx = 0.015708
approx = 1.8758
Elapsed time is 0.00030899 seconds.
```

Figure 3.10: Выводы

Как можем заметить, второй файл с векторизацией работает куда быстрее. А это значит, что лучше всего вместо циклов использовать операции над векторами.

На этом лабораторная работа закончена.

4 Выводы

Познакомился с методами работы с последовательностями, пределами, рядами.

5 Список литературы

Лабораторная работа №6

Лабораторная работа № 6. Введение в работу с Octave [Электронный ресурс].

2019. URL:https://esystem.rudn.ru/pluginfile.php/2372908/mod_resource/content/2/README.pdf