

Пределы, последовательности и ряды

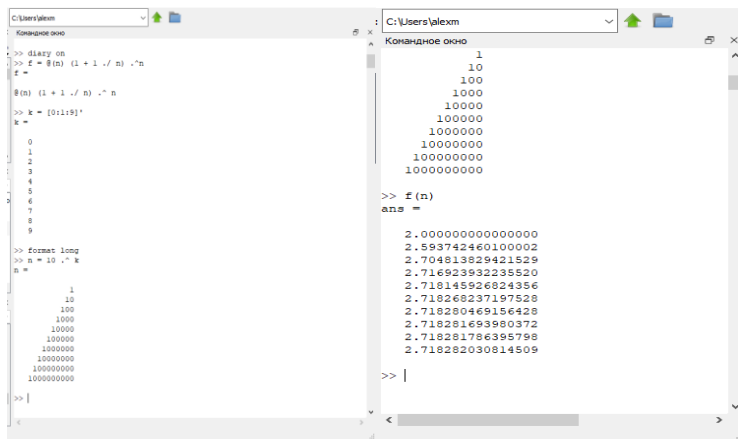
Милёхин Александр НПМмд-02-21

Цель работы

Научиться работать в Octave с пределами, последовательностями и рядами, а также научиться писать векторизованный программный код.

Пределы. Оценка

Определяем с помощью анонимной функции простую функцию. Создаём индексную переменную, возьмём степени 10, и оценим функцию.



```
C:\Users\alexm
Командное окно

>> 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

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

1
10
100
1000
10000
100000
1000000
10000000
100000000
1000000000

>> |

C:\Users\alexm
Командное окно

1
10
100
1000
10000
100000
1000000
10000000
100000000
1000000000

>> f(n)
ans =

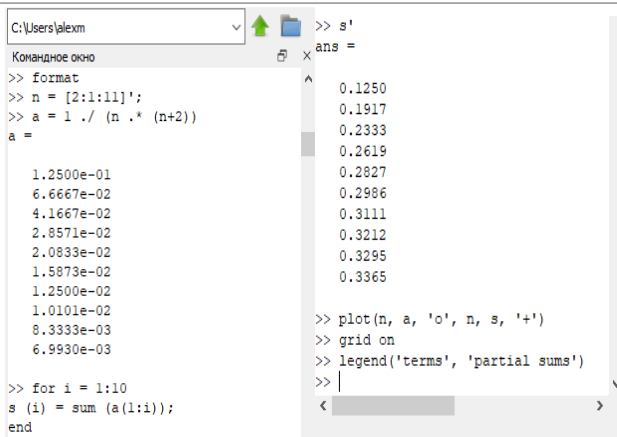
2.000000000000000
2.5937424460100002
2.704813829421529
2.716923932235520
2.718145926824356
2.718268237197528
2.718280469156428
2.718281693980372
2.718281786395798
2.718282030814509

>> |
```

Figure 1: Пределы. Оценка. Выполнение команд

Частичные суммы

Определим индексный вектор, а затем вычислим члены. После чего введем последовательность частичных сумм, используя цикл.



The image shows a MATLAB Command Window with the following code and output:

```
C:\Users\alexm
Командное окно
>> format
>> 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 2: Частичные суммы. Выполнение команд

Частичные суммы

Построенные слагаемые и частичные суммы представлены ниже.

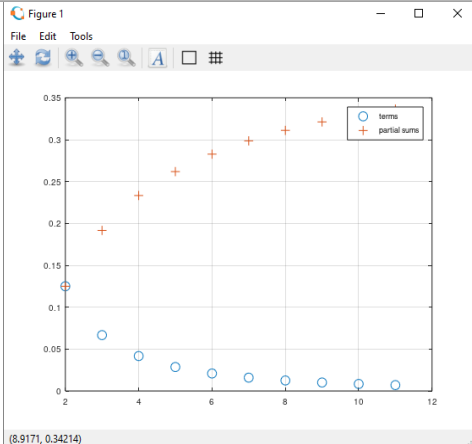
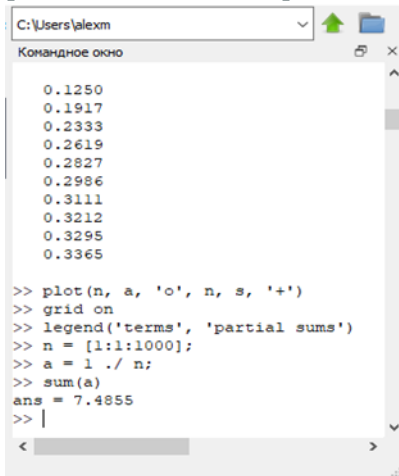


Figure 3: Построение слагаемых и частичных сумм

Сумма ряда

Найдём сумму первых 1000 членов гармонического ряда $1/n$.



The screenshot shows a Windows command prompt window titled "Командное окно" (Command Window) with the address bar set to "C:\Users\alexm". The window displays the output of several MATLAB commands. The first part shows a list of values: 0.1250, 0.1917, 0.2333, 0.2619, 0.2827, 0.2986, 0.3111, 0.3212, 0.3295, and 0.3365. Below these, the MATLAB commands and their outputs are shown: `>> plot(n, a, 'o', n, s, '+')`, `>> grid on`, `>> legend('terms', 'partial sums')`, `>> n = [1:1:1000];`, `>> a = 1 ./ n;`, `>> sum(a)`, and the final output `ans = 7.4855`.

```
C:\Users\alexm
Командное окно

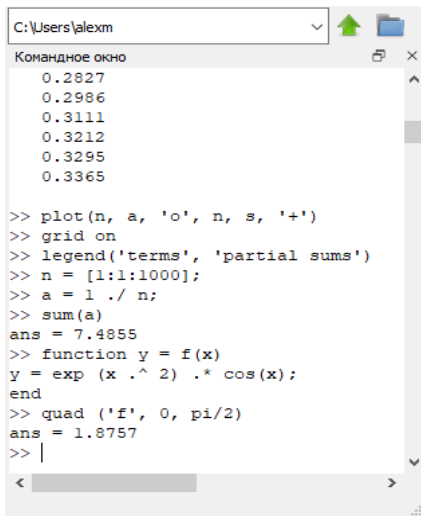
0.1250
0.1917
0.2333
0.2619
0.2827
0.2986
0.3111
0.3212
0.3295
0.3365

>> plot(n, a, 'o', n, s, '+')
>> grid on
>> legend('terms', 'partial sums')
>> n = [1:1:1000];
>> a = 1 ./ n;
>> sum(a)
ans = 7.4855
>> |
```

Figure 4: Сумма ряда

Вычисление интегралов

Численно посчитаем интеграл.



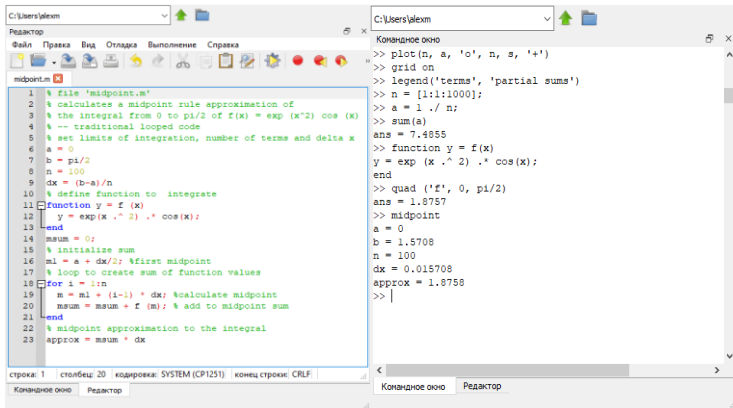
```
C:\Users\alexm
Командное окно
0.2827
0.2986
0.3111
0.3212
0.3295
0.3365

>> plot(n, a, 'o', n, s, '+')
>> grid on
>> legend('terms', 'partial sums')
>> n = [1:1:1000];
>> a = 1 ./ n;
>> sum(a)
ans = 7.4855
>> function y = f(x)
y = exp (x .^ 2) .* cos(x);
end
>> quad ('f', 0, pi/2)
ans = 1.8757
>> |
```

Figure 5: Интегрирование функции

Аппроксимирование суммами

Напишем скрипт для того, чтобы вычислить интеграл по правилу средней точки. Введём код в текстовый файл и назовём его `midpoint.m`. Запустим этот файл в командной строке.



The screenshot displays the MATLAB environment with two windows: 'Редактор' (Editor) and 'Командное окно' (Command Window).

Редактор (midpoint.m):

```
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 looped 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 midpoint
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 approx = msum * dx
```

Командное окно:

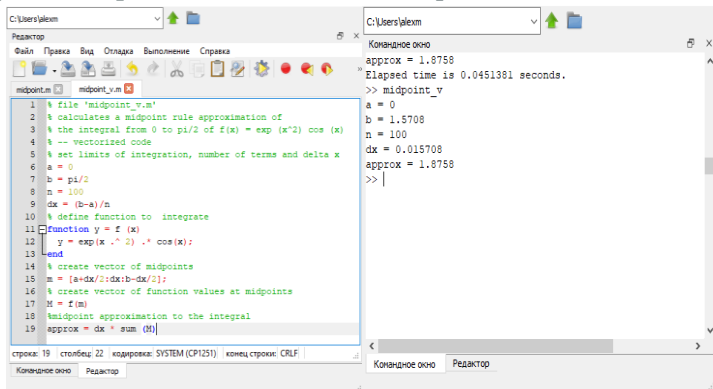
```
>> plot(n, a, 'o', n, b, '+')
>> grid on
>> legend('terms', 'partial sums')
>> n = [1:1:1000];
>> a = 1 ./ n;
>> sum(a)
ans = 7.4855
>> function y = f(x)
y = exp(x.^2) .* cos(x);
end
>> quad('f', 0, pi/2)
ans = 1.8757
>> midpoint
a = 0
b = 1.5708
n = 100
dx = 0.015708
approx = 1.8758
>> |
```

Строка: 1 столбец: 20 кодировка: SYSTEM (CP1251) конец строки: CRLF

Figure 6: Вычисление интеграла по правилу средней точки

Аппроксимирование суммами

Теперь напомним векторизованный код, не требующий циклов. Для этого создадим вектор x-координат средних точек. Запустим этот файл `midpoint_v` в командной строке.



The screenshot shows the MATLAB environment. The Editor window displays the file `midpoint_v.m` with the following code:

```
1 % file 'midpoint_v.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 % -- 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 values at midpoints
17 M = f(m);
18 %midpoint approximation to the integral
19 approx = dx * sum(M)
```

The Command Window shows the execution results:

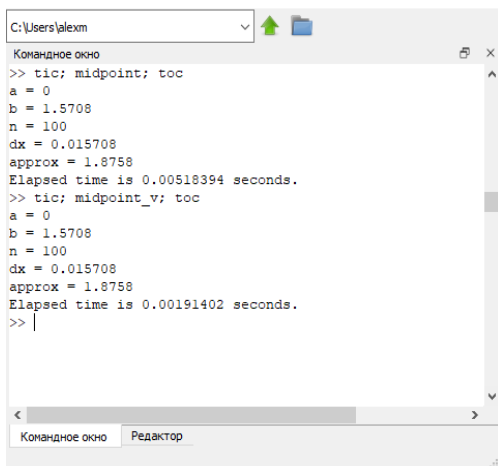
```
>> approx = 1.8758
Elapsed time is 0.0451381 seconds.
>> midpoint_v
a = 0
b = 1.5708
n = 100
dx = 0.015708
approx = 1.8758
>> |
```

At the bottom, the status bar indicates "строка: 19 столбец: 22 кодировка: SYSTEM (CP1251) конец строки: CRLF".

Figure 7: Векторизованный код программы

Аппроксимирование суммами

Запустим оба кода.



The screenshot shows a MATLAB Command Window with the following content:

```
C:\Users\alexm
Командное окно
>> tic; midpoint; toc
a = 0
b = 1.5708
n = 100
dx = 0.015708
approx = 1.8758
Elapsed time is 0.00518394 seconds.
>> tic; midpoint_v; toc
a = 0
b = 1.5708
n = 100
dx = 0.015708
approx = 1.8758
Elapsed time is 0.00191402 seconds.
>> |
```

At the bottom of the window, there are two tabs: "Командное окно" (Command Window) and "Редактор" (Editor).

Figure 8: Сравнение полученных результатов

Результат лабораторной работы

Я научился работать в Octave с пределами, последовательностями и рядами, а также научился писать векторизованный программный код. Более того, мне удалось определить, что векторизованный код работает существенно быстрее, чем код с циклами.

Спасибо за внимание