Шаблон отчёта по лабораторной работе

Турсунов Баходурхон Азимджонович

Содержание

Список иллюстраций

Список таблиц

Выполнение 6 лабораторной работы

Пределы, последовательности и ряды

- Octave - полноценный язык программирования, поддерживающий множество типов циклов и условных операторов.

- Расспишем простую функцию:

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

@(n) (1 + 1 ./ n) .^ n
```

(Рис 1)

- метод которую я использовал здесь называется анонимной функцией. Это хороший способ быстро определить простую функцию

1. Далее создал 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
```

(Рис 2)

- 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 возьмем степени 10, которые буду 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
   2.704813829421528
   2.716923932235594
   2.718145926824926
   2.718268237192297
   2.718280469095753
   2.718281694132082
   2.718281798347358
   2.718282052011560

>> format
```

(Рис 3)

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

Частичные суммы

1. Определим индексный вектор n от 2 до 11, а затем вычислим члены.

```
>> 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
```

(Рис 4)

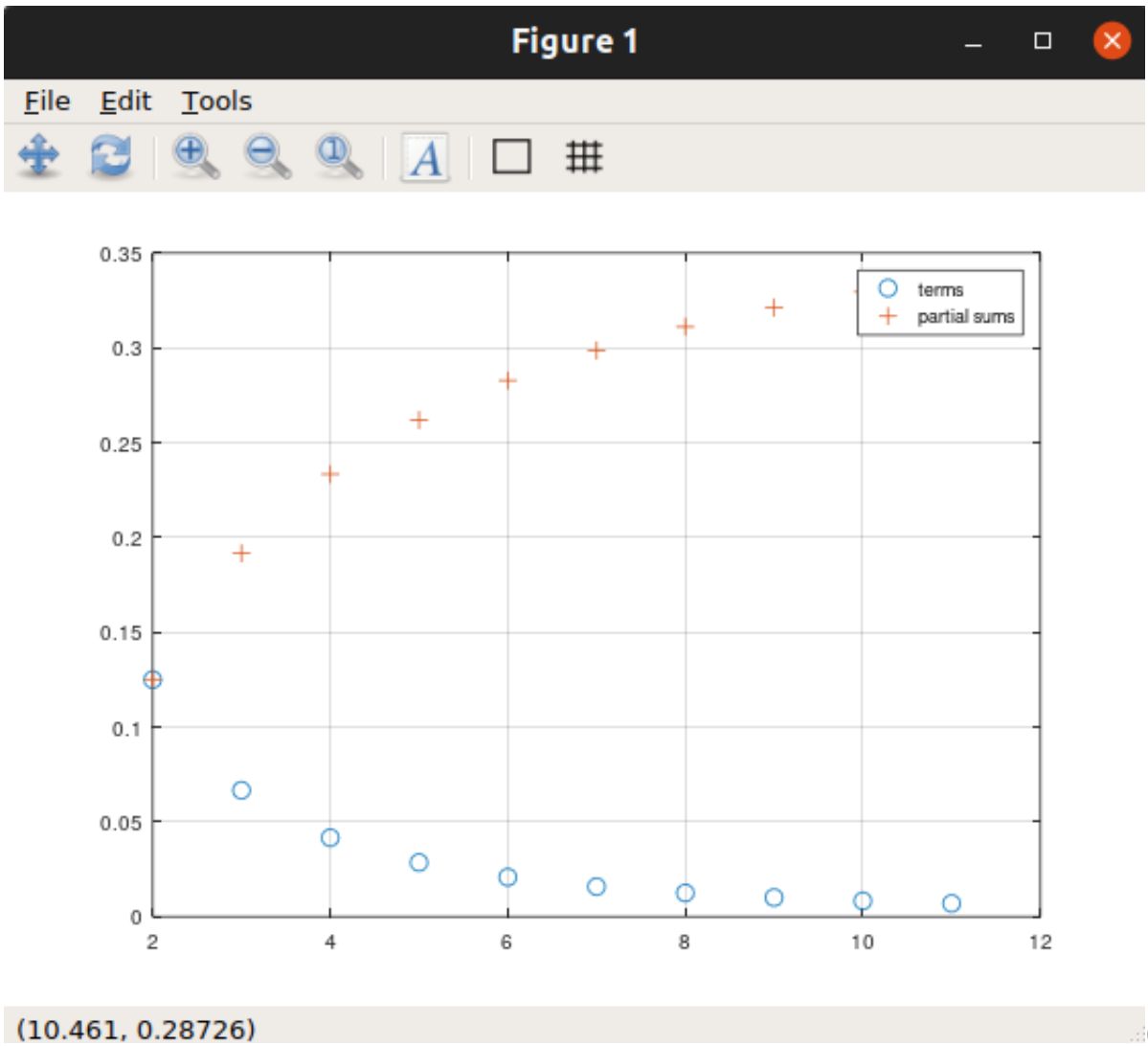
2. Если мы хотим знать частичную сумму, нам нужно написать sum(a). Если мы хотим получить последовательность частичных сумм, нам нужно использовать цикл. Мы будем использовать цикл for с индексом i от 1 до 10. Для каждошо i мы получим частичную сумму последовательности an от первого слагаемого до i-го слагаемого. На выходе получается 10-элементный вектор этих частичных сумм.

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

   0.1250
   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')
```

(Рис 5)



(Рис 6)

Сумма ряда

1. Найдем сумму первых 1000 членов гармоничного ряда. Нам нужно сгенерировать чекы как ряда вектор, а затем взять их сумму.

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

(Рис 6)

Численное интегрирование

Вычисление интегралов

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

1. Определяем функцию, чей интеграл мы будем считать, и считаем определенный интеграл командой quad.

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

(Рис 7)

Аппроксимирование суммами

1. Создал файл midpoint.m и записал туда код

```
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
```

(Рис 8)

2. Запустил файл и вот результат:

```
>> midpoint
a = 0
b = 1.5708
n = 100
dx = 0.015708
m1 = 7.8540e-03
approx = 1.8758
```

(Рис 9)

3. Далее создал еще один файл с название midpoint_v.m и записал туда код:

```
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)
```
(Рис 10)

4. Результат:

```
>> midpoint_v
a = 0
b = 1.5708
n = 100
dx = 0.015708
approx = 1.8758
```
(Рис 11)

5. Сравнил результаты выполнения каждой реализации:

```
>> tic; midpoint; toc
a = 0
b = 1.5708
n = 100
dx = 0.015708
m1 = 7.8540e-03
approx = 1.8758
Elapsed time is 0.00868607 seconds.
>>
>>
>> tic; midpoint_v; toc
a = 0
b = 1.5708
n = 100
dx = 0.015708
approx = 1.8758
Elapsed time is 0.000321865 seconds.
```

(Рис 12)

Вывод

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