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:

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

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.

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

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

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
```

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)`.

```
>> 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
```

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

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
```

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
```

Наконец, мы построим слагаемые и частичные суммы для $2 \leq n \leq 11$.

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

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;
>> sum (a)
ans =  7.4855
```

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
```

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.

Аппроксимирование суммами

Правило средней точки, правило трапеции и правило Симпсона являются общими алгоритмами, используемыми для численного интегри-

рования.

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.

Введите код в текстовом файле и назовите его `midpoint.m`.

```
% file 'midpoint.m'
% calculates a midpoint rule approximation of
% the integral from 0 to pi/2 of f(x) = exp (x^2) cos (x)
% -- traditional looped 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
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
```

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

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

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.

```
% 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)
```

Запустим его.

```
>> midpoint_v
a = 0
```

```
b =  1.5708
n =  100
dx =  0.015708
approx =  1.8758
```

Сравните результаты.
Сравните время выполнения для каждой реализации.

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

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

Задание

- Сделайте отчёт по лабораторной работе в формате Markdown.
- В качестве ответа просьба предоставить отчёты в 3 форматах: `pdf`, `docx` и `md` (в архиве, поскольку он должен содержать скриншоты, Makefile и т.д.)