Основы программирования в MATLAB

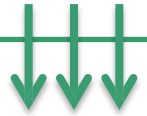
Лекция #6

Пустовалова О.Г.
доцент. каф. мат.мод.
ИММиКН ЮФУ

Содержание

Измерение времени

```
% Включить таймер
t1=tic();
% Матрица гильберта 1000 порядка
h=hilb(1000);
% Вывод времени, прошедшего
% с момента запуска таймера
t2=toc(t1)
```



```
t2 = 0.0272
```

ПРОГРАММИРОВАНИЕ В MATLAB

О программировании в MATLAB

- ✓ Язык программирования MATLAB является интерпретируемым, то есть выполняется сразу .
- ✓ Этап компиляции полной программы отсутствует.
- ✓ Высокая скорость выполнения обеспечивается наличием откомпилированного ядра
- ✓ Для выполнения программ необходима среда MATLAB
- ✓ Существуют компиляторы, преобразующие код MATLAB в C и C++

Язык программирования MATLAB

Язык высокого уровня для научно-технических расчетов

MATLAB поддерживает следующие виды программирования:

- Процедурное
- Операторное
- Функциональное
- Логическое
- Структурное или модульное
- Объектно-ориентированное
- Визуально-ориентированное

О именах файлов

М-файлы сценариев (script файлы) и функций должны иметь уникальные имена.

Длина имен не ограничивается, но только первый 31 символ учитывается при идентификации имени.

Если имя оказывается неуникальным, то соответствующий программный объект не исполняется, и выводится сообщение об ошибке.

Файлы-сценарии или файлы-скрипты

Файл-сценарий, именуемый также Script-файлом, является просто записью серии команд без входных и выходных параметров.

Он имеет такую структуру:

```
% Основной комментарий comment
% Дополнительный комментарий
Тело файла
```

Основной комментарий выводится при выполнении команд lookfor и help имя_каталога

Полный комментарий выводится при выполнении команды help Имя_файла

Файлы-сценарии или файлы-скрипты. Пример

```
% Построение поверхности Z=exp(-X^2)*cos(X*Y)
% meshgrid – задание сетки
% surf - поверхностьй
clc;
clear;
[X, Y] = meshgrid([-3:0.1:3]);
Z = X.*exp(-X.^2-Y.^2);
surf(X, Y, Z)
```

Знак % в комментариях должен начинаться с первой позиции строки.

В противном случае команда help name не будет воспринимать комментарий

Структура М-файла-функции

```
function [var1,var2,…]=file_name(Список параметров)
%Основной комментарий
%Дополнительный комментарий

Тело файла с любыми выражениями
var1=выражение
var2=выражение
...
```

Имя файла совпадает с именем функции!

Вызов функции:

```
[var1,va2,...]=file_name(Список параметров)
```

О глобальных и локальных переменных

Переменные в файлах-сценариях являются глобальными, а в файлах-функциях – локальными

```
function z=fun(x,y)
z=x^2+y^2;
```

Команда

```
global var1 var2…
```

позволяет объявить переменные модуля-функции глобальными.

Подфункции

Подфункции объявляются и записываются в теле основных функций и имеют идентичную им конструкцию.

```
function [ z ] = myF( x,y )
x=subF(x);
z=x+y;
end

function [ x ] = subF(x)
x=x^2;
end
```

```
myF(2,3)

ans = 7
```

Функция error

Для вывода сообщения об ошибке служит команда
error('Сообщение об ошибке')

```
function [ z ] = myF( x,y )
if y==0
    error(' Error - division by zero ')
else z=x/y;

end
```

```
myF(2,0)
Error using myF (line 3)
 Error - division by zero
```

```
myF(2,2)
ans = 1
```

Функция nargchk

Функция nargchk используется внутри m-файлов для проверки соответствия количества входных параметров

min и max

```
>> msg = nargchk(4,9,5)
```

→

```
msg = []
```

Число аргументов

```
>> msg = nargchk(4,9,2)
```

→

```
msg = Not enough input arguments.
```

Функция lasterr

Для вывода сообщения о последней произошедшей ошибке служит функция lasterr

```
>> q+2
Undefined function or variable 'q'.

>> lasterr

ans =

Undefined function or variable 'q'.
```

Функции с переменным числом аргументов

Вне тела m-файла функции

nargin и nargout

указывают, соответственно, количество входных или выходных аргументов, заданных пользователем.

```
>> nargin(@myF)    ⟶   ans = 2

>> nargout(@myF)   ⟶   ans = 1
```

Функции с переменным числом аргументов

Внутри тела m-файла функции

nargin и **nargout**

указывают, соответственно, количество входных или выходных аргументов, заданных пользователем.

Отрицательное число аргументов означает, что функция имеет переменное число аргументов;

Функции с переменным числом аргументов. nargin

```
%%
addme(2)          ⟶   4
addme(2,3)        ⟶   5
addme()           ⟶   0
%%
function c = addme(a,b)
    switch nargin
      case 2
        c = a + b;
      case 1
        c = a + a;
      otherwise
        c = 0;
     end
end
```

Количество входных аргументов может быть не больше двух.

Переменные varargin и varargout

```
function [  ] = myF( varargin )
 for i=1:nargin
     varargin(i)
end
```

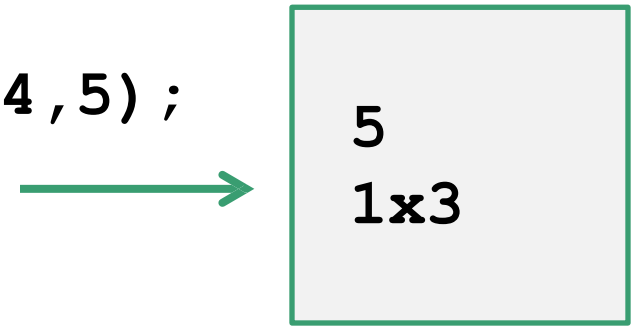
```
>> myF(1,2)  ⟶  ans = [1] ans = [2]

>> myF('A',3.14,rand(1))

⟶ ans = 'A' ans = [3.1400] ans = [0.9649]
```

Функции с переменным числом аргументов. varargin

```
%%
VarNum(1);
VarNum(1,2);
VarNum(1,2,3);
VarNum(ones(3),'some text',pi,3+2*i);
%%
function VarNum(varargin)
 disp("Кол-во входных параметров: " + nargin)
 celldisp(varargin)
end
```

Функции с переменным числом аргументов. varargin

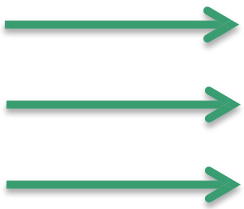
```
%%
defAndVarNumInputs(1,2,3,4,5);
%%
```




```
function defAndVarNumInputs(X, Y, varargin)
 disp("Всего параметров: " + nargin)
 formatSpec = "Доп. параметры.: %dx%d";
 str = compose(formatSpec,size(varargin));
 disp(str)
end
```

Функции с переменным числом аргументов. varargin

```
%%
addme(2)
addme(2,3)
addme(1,2,3)
%%
function c = addme(varargin)
    switch nargin
      case 2
        c = varargin{1} + varargin{2};
      case 1
        c = varargin{1};
      otherwise
        c = 0;
     end
end
```



Любое количество входных аргументов.

Переменные varargin и varargout

```
function varargout = vNumInAndOut(varargin)
  disp(['Кол-во входных: ' num2str(length(varargin))])
  disp(['Кол-во выходных: ' num2str(nargout)])
  for k = 1:nargout
    varargout{k} = k;
  end
end
```

```
[r]=varNumInputAndOutput(1,2,3)        → r=1
[r,s]=varNumInputAndOutput(1,2,3)      → r=1, s=2
[r,s,t]=varNumInputAndOutput(1,2,3)    → r=1, s=2, t=3
```

Анонимные функции

```
>> myf=@(x,y)x-y;
 myf(2,3)
```

```
ans =  -1
```

Анонимная функция состоит из одного выражения MATLAB и любого количества входных и выходных параметров.

Можно определить анонимную функцию прямо в командной строке MATLAB или в пределах функции или скрипта.

УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

Цикл for end

```
for i=1:5
    i^2
end
```

```
for i=0:0.25:1
    i^2
end
```

```
for i=1:3
    for j=1:3
        A(i,j)=i^j
    end
end
```

```
A =
     1     1     1
     2     4     8
     3     9    27
```

Цикл for end

```
A =

     1     2
     3     4
```

```
A=[1 2; 3 4]

for x=A
    x
end
```

```
x =   1
      3

x =   2
      4
```

Множественный выбор switch end

```
x=5

switch x
    case {1,2,12}
        disp('winter')
    case {3,4,5}
        disp('spring')
    case {6,7,8}
        disp('summer')
    case {9,10,11}
        disp('autumn')
end
```

Цикл while end

```
s=0

while s<5
    s=s+1
end
```

Оператор continue

```
k=0;
while k<4
      k=k+1; if
      (k==2)
            continue
      else
            prod(1:k)
      end
end;
```

Оператор break

```
k=0;

while k<6
      k=k+randi(2,1)
      if (k==4)
            break
      end
end;
```

Операторы try и cath

```
n=randi(2,1)
A=randi(5,n)
b=randi(2,2,1)

try
     linsolve(A,b)
catch
     msgstr = lasterr
End
```

Спасибо за внимание!

Переменные varargin и varargout

```
function [s,varargout] = myF(x)
nout = max(nargout,1) - 1;
% nargout – количество выходных аргументов
s = size(x);
for k=1:nout
    varargout{k} = s(k);
end
```

```
[s,rows,cols] = myF(rand(4,5,2))
```

```
s = 4     5     2
rows = 4    cols = 5
```