

Файл-функции и скрипты

Основы MATLAB

Юдинцев В. В.

Кафедра теоретической механики

13 сентября 2024 г.



САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

- 1 Функции пользователя
- 2 Файлы-функции и скрипты
- 3 Файл-скрипты
- 4 Файл-функции

Функции пользователя

Способы задания функции

- inline функции;
- анонимные функции;
- файл-функции.

inline-функция

```
1 >> f1 = inline('x1^2+x2^2','x1','x2');  
2 >> f1(4,2)  
3 ans =  
4     20
```

Переменные из рабочей области недоступны.

Анонимная функция

```
1 >> f = @(x1,x2) x1^2+x2^2;  
2 >> f(4,2)  
3 ans =  
4     20
```

Переменные из рабочей области доступны, но рассматриваются как константы:

```
1 >> a=2;  
2 >> f = @(x1) a*x1;  
3 >> f(1)  
4 ans =  
5     2  
6 >> a=1; f(1)  
7 ans =  
8     2
```

Файлы-функции и скрипты

Типы файлов

- * `.m` (текстовые) содержат тексты программ, определения функций.
- * `.mat` (бинарные) содержат значения переменных.
- * `.mex` (бинарные). МЕХ-файлы – динамически подключаемые библиотеки

- Любую последовательность команд в MATLAB можно оформить в виде `m` файла.
- По умолчанию все переменные, объявленные внутри файл-скрипта, являются глобальными.

- Файл-функция содержит определение одной или нескольких функций.
- По-умолчанию все переменные, объявленные внутри файл-функции, являются локальными.
- Файл-функция является самостоятельным программным модулем, который связан с другими модулями и головной программой через входные и выходные параметры.

Создание m-файлов

- При создании файл-функций и файл-скриптов следует избегать перекрытия имён других функций.
- Для проверки имени можно использовать функцию `exist`:

```
1 >> exist atan
2 ans =
3     5
```

- Если имя не занято, то функция `exist` возвращает 0 (5 – если имя занято встроенной функцией).

Файл-скрипты



- Файл-скрипт не имеет входных и выходных аргументов.
- Работает с данными из рабочей области.
- Все переменные, объявленные в файл-скрипте, являются глобальными.
- В процессе выполнения не компилируются.
- Представляют собой зафиксированную в виде файла последовательность операций.

Структура файл-скрипта

Файл `fscript.m`

```
1  % Описание кода ,  
2  % которое можно увидеть  
3  % напечатав help fscript в окне команд  
4  x=1:0.1:10;  
5  y=sin(x);
```

Выполнение файл-скрипта

- Вызов из командной строки (command window)
- Запуск из редактора при помощи сочетания  
- Файл-скрипт можно для удобства разделить на ячейки (секции) при помощи удвоенного знака %%

%% секция 1

код ...

%% секция 2

код ...

- Код в каждой секции может быть выполнен при помощи

Файл-функции

Объявление функции

File New Function

```
1 % комментарии к функции ,  
2 % которые будут выводиться  
3 % по команде  
4 % help func_name  
5 function [out1 , out2] = func_name (in1 , in2)  
6 %  
7 % тело функции  
8 %  
9 out1 = ...  
10 out2 = ...  
11
```

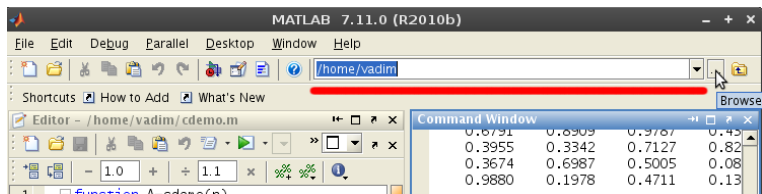
Глобальные переменные в функциях

- По умолчанию все переменные внутри функции являются локальными.
- Для того, чтобы несколько функций использовали одну переменную, её необходимо объявить глобальной.

```
1 function res = func_name(in1 , in2)
2   global G;
3   res=G*in1+in2 ;
```

Выполнение функций и файл-скриптов

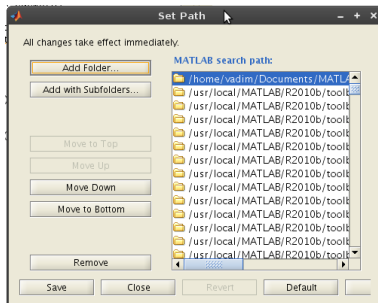
Имя файла и имя объявленной в нем функции предпочтительно делать одинаковыми. Каталог, в котором содержатся вызываемые функции, должен быть текущим или добавлен в пути поиска.



Выполнение функций и файл-скриптов

File

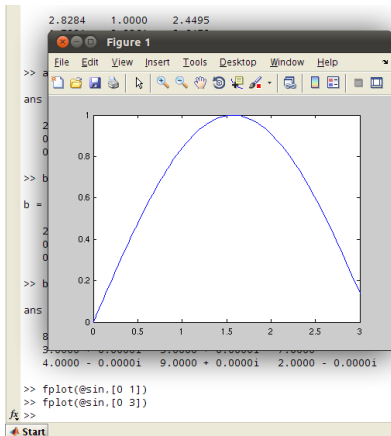
Set Path



Оператор return

- Функция прекращает работу после выполнения последнего оператора.
- Принудительно завершить функцию можно оператором `return`.

Быстрое построение графика функции



Функция `fplot`

```
1 fplot('myfun',[0 4])
2 fplot(@myfun,[0 4])
```

первый аргумент: имя или ссылка на функцию, второй аргумент: диапазон изменения аргумента функции для построения графика

Внутренние функции

- Файл-функция вместе с определением основной функции может содержать определения вспомогательных функций, доступных к вызову только из основной функции.

```
1 function f = myfun(x)
2     f1=infun(x);
3     f=f1+cos(x);
4 % Внутренняя функция
5 function f = infun(x)
6     a=3;
7     f=sin(x*3);
```

- Переменные, используемые в подфункциях **локальные**.

Вложенные функции

- Вложенная функции определяется в теле основной функции.
Файл `myfun.m`:

```
1 function f = myfun(x)
2     f1=infun(x);
3     f=f1+cos(x);
4     function f = infun(x)
5         f=sin(x);
6     end
7 end
```

- Из вложенной функции `infun` доступны локальные переменные всех функций верхнего уровня и наоборот.

Функции с переменным количеством аргументов

- `varargin` – список ячеек с параметрами функции;
- `length(varargin)` – количество переданных аргумента;
- `varargin{1}` – первый аргумент;
- `varargin{2}` – второй аргумент.

```
1 function res = fun(varargin)
2     if length(varargin) < 2
3         error('Недостаточное количество аргументов');
4     end
5     ...
```

Функции с переменным количеством аргументов

Аргумент `varargin` может быть указан после перечня обязательных аргументов

```
1 function res = fun(a1,a2,varargin)
2     if length(varargin)<4
3         error('Недостаточное количество аргументов');
4     end
5     ...
```

Функция в качестве аргумента

Использование функции `feval`

Передача имени функции строкой:

```
1 >> x = 1;  
2 >> feval('sin',x)  
3 ans =  
4     0.8415
```

Передача ссылки на функцию (это работает быстрее):

```
1 >> feval(@sin,x)  
2 ans =  
3     0.8415
```