M-файлы

- **M-сценарии** (m-файлы скрипты) – последовательность команд без входных и выходных параметров; оперирует с данными из рабочей области; результаты сохраняются в рабочей области и могут далее использоваться
- **M-функции** – используют входные и выходные аргументы; имеют внутренние локальные переменные

File – New – M-File – создание M-файлов

Структура М-функции

- **Строка определения функции** задаёт имя функции и входные и выходные аргументы, их локальные имена

  function y = function_name (u,v,w)
- **Первая строка комментария** определяет назначение функции; выводится на экран командами: lookfor или

  help < function_name>
- **Основной комментарий**
- **Тело функции** – программный код

М-функция записывается в файл с тем названием, что и функция и с расширением m.

Если выходных параметров больше, они указываются в квадратных скобках: function [u,v,w] = sphere(z,theta,rho)

Подфункции

- М-функции могут содержать коды более, чем одной функции. Первая функция в файле – основная, вызываемая по имени М-файла. Другие – подфункции, видимые только для основной функции и других подфункций этого файла. Они имеют свой заголовок, следуют друг за другом непрерывно и могут вызываться в любом порядке. Основная функция выполняется первой.

Пример

Функция, находящая среднее значение и медиану для элементов вектора u:

```
 function [avr,med] = newstatus(u)    %  основная функция
 n = length(u);
 avr = mean(u,n);
 med = median(u,n);
function a = mean(v,n)  %  подфункция
%  вычисление среднего
 a = sum(v)/n;
 function m = median(v,n)  % подфункция
%  вычисление медианы
 w = sort(v);
If rem(n,2) == 1
     m = w((n+1)/2;
 else
     m =  (w(n/2)+w(n/2+1))/2;
end
```

Частные функции

Частный каталог – подкаталог с именем private родительского каталога. M-файлы частного каталога доступны только M-файлам родительского каталога и вне него не видимы (создание собственных версий функций, сохраняя оригиналы в другом каталоге. MATLAB просматривает частный каталог раньше каталогов стандартных функций системы MATLAB и в первую очередь использует функции из частного каталога.

Вызов функции

- Можно вызвать из командной строки или из других М-файлов, указав все атрибуты (входные аргументы в круглых скобках, выходные в квадратных). При появлении нового имени функции или переменной MATLAB проверяет:
- Является ли новое имя именем переменной
- Является ли это имя именем подфункции данного М-файла
- Является оно именем частной функции в каталоге private
- Является оно именем функции в пути доступа системы MATLAB.

В случае дублирования имён используется первое имя в соответствии с указанной иерархией. Допускается переопределять функцию.

При вызове М-функции система транслирует её в псевдокод и загружает в память (нет повторного синтаксического анализа). Псевдокод остаётся в памяти пока не завершён сеанс работы или не использована команда clear.

Команда clear

- **clear <имя функции>** - удаление указанной функции из рабочей области
- **clear functions** – удаление всех откомпилированных программ (их можно сохранить для последующей работы командой pcode myfunc – выполняется синтаксический анализ M-файла myfunc.m и сохраняется p-код в файле с именем myfunc.p, однако справка об этой функции уже не доступна; применимость – анализ большого числа M-файлов или желание скрыть алгоритмы, реализованные в M-файле)
- **clear all** – удаление всех программ и данных

Рабочая область функции – выделяется дополнительная область памяти, не пересекающаяся с рабочей область системы

Проверка количества аргументов:

входных – функцией **nargin**
выходных – функцией **nargout**

Они спользуются для изменения хода вычислений.

Пример:
```
function c = testarg(a,b)
if (nargin == 1)
    c = a.^2;
elseif (nargin == 2)
    c = a+b;
 end
```

Если при обращении к М-функции выходной аргумент не указан, по умолчанию выводится первый аргумент.

Произвольное количество аргументов

– в списках аргументов ставятся переменные **varargin** и **varargout**.

Переменная varargin должна быть последней в списке входных аргументов после всех обязательных, а переменная varargout – последней в списке выходных аргументов.

Локальные и глобальные переменные

Команда **global**

Глобальные переменные могут быть изменены из командной строки, без редактирования М-файла. Для работы с ними необходимо:

- объявить переменную как глобальную во всех М-функциях, где необходима эта переменная
- в каждой функции использовать команду global перед первым появлением переменной; эту команду рекомендуется использовать в начале М-файла.

Режим ячейки

MATLAB позволяет разделять М-файл-сценарий на части – ячейки. Чтобы запустить новую ячейку надо вставить строку комментария, которая служит заголовком ячейки, начав её с %%.

Откройте М-файл в модуле **Editor**, команда **Cell - Enable Cell Mode** (Ячейка-Включить режим ячейки) – откроется вторая панель инструментов для работы с операторами ячейки.

Вычисление текущей ячейки – команда меню **Cell - Evaluate Current Cell**

Команда **Evaluate Cell and advance**– вычислить ячейку и далее.

Создание дополнительных ячеек – команда **Insert Cell Divider**

Операторы MATLAB

- Арифметические операторы (арифметические выражения, вычисления)
- Операторы отношения (сравнение аргументов)
- Логические операторы (логические выражения)

Уровни приоритета арифметических операторов

1. Поэлементное транспонирование (.’), поэлементное возведение в степень (.*), сопряжение матрицы (‘), возведение матрицы в степень(^)
2. Унарный плюс(+) и унарный минус (-)
3. Поэлементное умножение массивов(.*), правое деление массивов (./) и левое (.\), умножение матриц (*), решение систем линейных уравнений операции (/) и (\)
4. Сложение (+) и вычитание массивов (-)
5. Оператор (:)

Внутри каждого уровня операторы имеют равный приоритет и вычисляются в порядке следования слева направо. Порядок вычислений может быть изменён скобками.

При работе с массивами , один из операндов скаляр, то он расширяется до размеров второго операнда и заданная операция применяется к каждому элементу.

Операторы отношения

- < - меньше – функция lt()
- <= - меньше или равно – функция le()
- > - больше – функция gt()
- >= - больше или равно – функция ge()
- == - равно - функция eg()
- ~= - не равно – функция ne()

- поэлементное сравнение двух массивов равных размерностей
- когда один операнд – скаляр, он сравнивается с каждым элементов второго операнда (логическое значение 1 и 0)
- Операторы отношения обычно используются для изменения последовательности операторов программы, поэтому обычно используются в теле операторов if, for, while, switch.
  Их приоритет ниже арифметических, но выше

Логические операторы

- & - массив: 1- для каждого местоположения, в котором оба элемента имеют значение true (отличны от нуля) и 0 – для всех остальных элементов; функция and()
- | - масив: 1- для каждого местоположения, в котором хотя бы один элемент имеет значение true (отличен от нуля) и 0 – для всех остальных элементов; функция or()
- ~ - логическое отрицание для каждого элемента входного массива, A; функция not()
- xor – массив: 1- для каждого местоположения, в котором только один элемент имеет значение true (отличен от нуля) и 0 – для всех остальных элементов

Пример: A = [0 1 1 0 1]; B = [1 1 0 0 1];

A&B = 01001

A|B = 11101

~A = 10010

xor(A,B)=10100

Управление последовательностью исполнения операторов

- **If** – оператор условия, в сочетании с оператором else и elseif выполняет группу инструкций в соответствии с некоторыми логическими условиями;
- **switch** – оператор переключения, в сочетании с операторами case и otherwise выполняет различные группы инструкций в зависимости от значения некоторого логического условия;
- **while** – оператор условия, выполняет группу инструкций неопределенное число раз, в соответствии с некоторым логическим условием завершения;
- **for** – оператор цикла, выполняет группу инструкций фиксированное число раз;
- **continue** – передаёт управление к следующей итерации цикла for или while, пропуская оставшиеся инструкции в теле цикла;
- **break** – прерывает выполнение цикла for или while;
- **try…catch** – изменяет управление потоком данных при обнаружении ошибки во время выполнения;
- **return** – возвращение к функции вызова.

Все операторы включают оператор end, чтобы указать конец блока, в

Оператор условия

if…else…elseif…end

Первая форма:	**Вторая форма:**	**Третья форма:**
If логическое_выражение команда end	If логическое_выражение команда else команда End	If логическое_выражение команда elseif команда else команда end

Оператор переключений
switch…case…otherwise…end

```
Switch   выражение (скаляр или строка)
  case value1
    команды  % Исполняются, если выражение есть value1
  case value2
    команды  % Исполняются, если выражение есть value2
   . . .
otherwise
    команды  % Исполняются, если не обработана ни одна
                 % из предыдущих групп case
End
```

Оператор цикла с неопределённым числом операций *while…end*

```
while      выражение
           команды
End
```

Оператор цикла с определённым числом операций *for…end*

```
for index = start:increment:end
           команды
end
```

Построение двумерных графиков

1. Функция **ezplot –** может работать

со строкой:

>> ezplot('x^2+x+1', [-2 2])

символьным выражением:

>> syms x, ezplot('x^2+x+1', [-2 2])

или анонимной функцией:

>> ezplot(@(x) x.^2+x+1, [-2 2])

- пример построения графика функции $y = x^2+x+1$ в интервале от -2 до 2.

Название рисунка:

- в Командном окне командой:

>> **title ‘A Parabola’**

- в окне рисунка командой **Edit – Axes Properties**

Ограничение диапазонов:

>> **axis** ([-1 2 0 3])

Первые два числа – это диапазон горизонтальной оси, вторые два числа – диапазон вертикальной оси.

Изображение квадрата графика:

>> **axis tight** - автоматически задаются диапазоны, включающие график полностью:

>> axis tight,ezplot('x^2+x+1')

Построение нескольких кривых:

команды **hold on** (наложение двух и более чертежей) и **hold off** (отмена предыдущей команды).

Пример

```
>> ezplot('exp(-x)', [0 10])
>> hold on
>> ezplot('sin(x)', [0 10])
>> hold off
>> title 'exp(-x) and sin(x)'
```

2. Функция **plot** – работает с векторами числовых данных

plot (x1, y1, s1, x2, y2, s2, x3, y3, s3, …),

x1, x2, x3 – аргументы функций,

y1, y2, y3 – имена функций или массивов значений ординат функций,

s1, s2, s3 – определяют тип и цвет линий графика, а также символ, отображающий точки на графике

Команды для подписи осей графика: **xlabel(text), ylabel(text)**

Команда для подписи графика: **title(text)**

Вывод легенд на график: **legend(text1,text2,…)**

Пример: >> x=0:0.01:10; plot(x,exp(-x),x,sin(x))

Задание цвета линий графиков:
plot (x1, y1, ‘g’) – линия зелёного цвета

Символ цвета	Цвет графика	Символ цвета	Цвет графика
y	Желтый (от англ. *yellow*)	g	Зеленый (от англ. *green*)
m	Малиновый (от англ. *magenta*)	b	Синий (от англ. *blue*)
c	Цианновый (от англ. *cyan*)	w	Белый (от англ. *white*)
r	Красный (от англ. *red*)	k	Черный (от англ. *black*)

Задание стиля линий графиков: сплошная линия (по умолчанию управляющий символ - тире), пунктирная линия (двоеточие), штрих-пунктирная линия (тире и точка), штриховая линия (два тире). Управляющие символы, определяющие стиль линии, задаются третьим параметром вместе с цветом). Порядок символов любой.

plot (x1, y1, ‘g:’) или **plot (x1, y1, ‘:g’)**

Задание типа маркера

Символ	Маркер	Символ	Маркер
.	Жирная точка	d	Ромбик
o	Кружок	v	Треугольник вершиной вниз
x	Косоугольный крестик	^	Треугольник вершиной вверх
+	Прямоугольный крестик	<	Треугольник вершиной влево
*	Восьмиконечная снежинка	>	Треугольник вершиной вправо
s	Квадратик	p	Пятиконечная звезда
none	Отсутствие маркера	h	Шестиконечная звезда

Если указан тип маркера, но не указан стиль линии, то табличные точки маркером метятся, но прямыми линиями не соединяются.

plot (x1, y1, ‘s--’) – маркеры квадратики на штриховой линии

Контурные чертежи

- Контурный чертёж функции с двумя переменными (это кривые уровня функции) создаётся командами **meshgrid** и **contour**.

meshgrid создаёт сетку из точек в прямоугольной области с заданными интервалами, а **contour** её использует для создания контурного чертежа в заданной области

- команда **ezcontour**

Пример

Построить контурный чертёж функции $x^2 + y^2$

1. >> contour(x,y,x.^2+y.^2); axis square

   или

   >> contour(x,y,x.^2+y.^2,[1 2 3]); axis square

2. >> ezcontour('x^2+y^2',[-3,3],[-3,3]); axis square

Пример

Построить графики функций $y=e^{sin(x)}cos(x)$, $z=e^{sin2(x)}$ на интервале $x \in [-\pi, \pi]$:

```
>> x = pi: pi/20: pi;
>> y = exp(sin(x)).*cos(x);
>> z = exp(sin(x).^2);
>> xlabel (“X”); ylabel (“Y”); title (“Графики функций”); legend(“Y”,’Z’);
```

Кривые в трёхмерном пространстве

- команда **ezplot3**
- команда **plot3**

Пример

Начертить спираль с координатами $x=\cos(2\pi z)$, $y=\sin(2\pi z)$

```
>> ezplot3('cos(2*pi*t)', 'sin(2*pi*t)', 't', [-2 2])
>> t=-2:0.01:2; plot3(cos(2*pi*t), sin(2*pi*t), t)
```

Поверхности в трёхмерном пространстве

- команда **mesh** (воспроизводит прозрачную сетчатую поверхность)
- команда **surf** (воспроизводит непрозрачную затенённую поверхность)
- сокращённые команды **ezmesh** и **ezsurf**

Пример

Построить график функции $z=x^2 - y^2$

```
>> [x,y]=meshgrid(-2:0.1:2, -2:0.1:2);

>> z=x.^2-y.^2; mesh(x,y,z)

>> z=x.^2-y.^2; surf(x,y,z)

>> ezmesh('x^2-y^2', [-2,2,-2,2])
```

Задания

1. Изучите листинг и выполнение прилагающихся программ.
2. Постройте графики следующих функций:

a) $Y=x^3-x$ для $-4 \le x \le 4$

b) $Y=\sin(1/x^2)$ для $-2 \le x \le 2$

c) $Y=\tan(x/2)$ для $-\pi \le x \le \pi$ и $-10 \le y \le 10$ (сначала сделайте чертёж, потом используйте команду axis)

d) $Y = e^{-x^2/2}$ и $y=x^4-x^2$ для $-1.5 \le x \le 1.5$ (на одной и той же координатной сетке)