Казанский национальный исследовательский
технический университет им. А. Н. Туполева-КАИ

Институт радиоэлектроники и телекоммуникаций

Кафедра радиоэлектронных и телекоммуникационных систем

Методические указания к лабораторной работе
«**Элементы программирования в среде MATLAB**»

по дисциплине
«Прикладные информационные технологии»

Казань, 2015 г.

Цель работы: знакомиться с элементами программирования в системе MATLAB, Научиться работать со скрипт-файлами и скрипт функциями

1.Функции пользователя

Хотя в ядро MATLAB последних версий встроено около тысячи операторов и функций, входящих как во входной язык MATLAB, так и в его язык программирования, пользователю всегда может понадобиться та или иная функция, простая или сложная, отсутствующая в ядре. Язык программирования систем MATLAB предоставляет ряд возможностей для задания функций пользователя.

Одна из таких возможностей заключается в применении функции *inline* , аргумент которой надо задать в апострофах. В приведенном ниже примере задана функция двух переменных - суммы квадратов *sin(x)* и *cos(y)*:

```
>> sc2=inline(' sin (x) .^2+cos(y) .^2')
sc2 =
Inline function:
sc2(x,y) = sin (x) .^2+cos(y) .^2
```

Можно также создать так, называемую, handle-функцию с помощью оператора @:

```
>> fh=@sc2;
```

К такой функции можно обращаться с помощью функции исполнения функций *feval (f h , x , y) :*

```
>> feval(fh,1,2)
y=
        0.8813
ans =
        0 .8813

>> feval(fh,2,1)
y=
        1.1187
ans =
         1.1187
```

2. Управляющие структуры

Язык программирования MATLAB имеет стандартные средства для задания управляющих структур, например циклов и условных выражений, а также двух основных программных объектов: скрипт-файлов и поименованных m-файлов-функций. Рассмотрим кратно их организацию.

Условный оператор *if* в общем виде записывается следующим образом:

```
if Условие
        Инструкции_1
elseif Условие
        Инструкции 2
else
        Инструкции 3
end
```

Эта конструкция допускает несколько частных вариантов. В простейшем, типа if . . . end

```
if Условие
    Инструкции
end
```

пока `Условие` возвращает логическое значение 1 (то есть «истина»), выполняются `Инструкции`, составляющие тело структуры `if...end`. При этом оператор `end` указывает на конец перечня инструкций. Инструкции в списке разделяются оператором, (запятая) или ; (точка с запятой). Если `Условие` не выполняется (дает логическое значение 0, т. е. «ложь»), то `Инструкции` также не выполняются.

Циклы типа f o r . . .end обычно используются для организации вычислений с заданным числом повторяющихся циклов. Конструкция такого цикла имеет следующий вид:

```
for var=Выражение, Инструкция, . . ., Инструкция end
```

`Выражение` чаще всего записывается в виде `s:d:e`, где `s` — начальное значение переменной цикла var, d — приращение этой переменной и `e` — конечное значение управляющей переменной, при достижении которого цикл

завершается. Возможна и запись в виде s : e (в этом случае d=1). Список выполняемых в цикле инструкций завершается оператором end.

Хорошо известный цикл типа while выполняется до тех пор, пока выполняется Условие:

```
while Условие
     Инструкции
end
```

Для осуществления множественного выбора (или ветвления) используется конструкция с переключателем типа switch:

```
switch switch_Выражение
     case сазе_Выражение
     Список инструкций
     case {сазе_Выражение1, Сазе_выражение2, сазе_Выражение3,...}
     Список инструкций
     ...
     otherwise,
     Список_инструкций
end
```

Если выражение после заголовка switch имеет значение одного из выражений case_Выражение., то выполняется блок операторов case, в противном случае — список инструкций после оператора otherwise. При выполнении блока case исполняются те списки инструкций, для которых case_Выражение совпадает со switch_Выражением. Обратите внимание на то, что case_Выражение может быть числом, константой, переменной, вектором ячеек или даже строчной переменной.

В управляющих структурах, в частности в циклах for и while , часто используются операторы, влияющие на их выполнение.

Так, оператор break может использоваться для досрочного прерывания выполнения цикла. Как только он встречается в программе, цикл прерывается.

Пример:

```
for i=1:10 i,
     if i==5 break,
     end,
end;
```

Оператор continue передает управление в следующую итерацию цикла, пропуская операторы, которые записаны за ним, причем во вложенном цикле он передает управление на следующую итерацию основного цикла

Пример:

```
for i=1:10 i,
     if i==5
       continue,
     end
     x=i;
end
```

Оператор `return` обеспечивает нормальный возврат в вызывающую функцию или в режим работы с клавиатурой. Пример применения оператора `return`:

```
function d = det(A)
%DET det(A) is the determinant of A.
if isempty(A)
d = 1;
     return
else
  ...
end
```

В данном примере если матрица A пустая, будет выведано значение 1, после чего управление будет передано в блок `else. . .end`.

3. Файлы-сценарии и файлы–функции

Файл-сценарий, именуемый также Script-файлом *(скрипт-файлом),* является просто поименованной записью пользователем серии команд без входных и выходных параметров. Он имеет следующую структуру:

```
%Основной комментарий
%Дополнительный комментарий
Тело файла с любыми выражениями
```

Важны следующие свойства файлов-сценариев:

- они не имеют входных и выходных аргументов;
- работают с данными из рабочей области;
- в процессе выполнения не компилируются;
- представляют собой зафиксированную в виде файла последовательность

операций, полностью аналогичную той, что используется в сессии.

Основным комментарием (в фирменной документации он назван H1) является первая строка текстовых комментариев, а дополнительным — последующие строки. Основной комментарий выводится при выполнении команд `lookfor` и `help имя_каталога`. Полный комментарий выводится при выполнении команды `help Имя_файла`. Рассмотрим следующий файл-сценарий:

```
% Plot with color red
% Строит график синусоиды линией красного цвета
% с выведенной масштабной сеткой в интервале [xmin,xmax]
x=xmin:0.1:xmax;
plot(x,sin(x),'r' )
grid on
```

Первые три строки здесь — это комментарий, остальные — тело файла. Обратите внимание на возможность (не рекомендуемую) задания комментария на русском языке. Знак % в комментариях должен начинаться с первой позиции строки.

Обратите внимание на то, что такой файл нельзя запустить без предварительной подготовки, сводящейся к заданию значений переменным xmin и xmax, использованным в теле файла. Это следствие первого свойства файлов сценариев — они работают с данными из рабочей области. Переменные, используемые в файлах сценариях, являются глобальными, то есть они действуют одинаково в командах сессии и внутри программного блока, которым является файл-сценарий. Поэтому заданные в сессии значения переменных используются и в теле файла. Имена файлов-сценариев нельзя использовать в качестве параметров функций, поскольку файлы-сценарии не возвращают значений.

М-файл-*функция* является типичным полноценным объектом языка программирования системы MATLAB. Одновременно он является полноценным модулем с точки зрения структурного программирования, поскольку содержит входные и выходные параметры и использует аппарат

локальных переменных. Структура такого модуля с одним выходным параметром выглядит следующим образом:

```
function var = f_name (Список параметров)
%Основной комментарий
%Дополнительный комментарий
Тело файла с любыми выражениями
var=выражение
```

М-файл-функция имеет следующие свойства:

- он начинается с объявления function, после которого указывается имя переменной var — выходного параметра, имя самой функции и список ее входных параметров;
- функция возвращает свое значение и может использоваться в виде name (Список_параметров) в математических выражениях;
- все переменные, имеющиеся в теле файла-функции, являются локальными, то есть действуют только в пределах тела функции;
- файл-функция является самостоятельным программным модулем, который общается с другими модулями через свои входные и выходные параметры;
- правила вывода комментариев те же, что у файлов-сценариев;
- файл-функция служит средством расширения системы MATLAB;
- при обнаружении файла-функции он компилируется и затем исполняется, а созданные машинные коды хранятся в рабочей области системы MATLAB.

Последняя конструкция var=выражение вводится, если требуется, чтобы функция возвращала результат вычислений.

Циклы for часто используются для реализации итерационных алгоритмов с заданным числом итераций. Рассмотрим классический пример — генерацию последовательности чисел Фибоначчи. Первые два числа f_1 и f_2 равны 1, а по последующие определяются как сумма двух предыдущих чисел, т.е.

$f_{i+2} = f_{i+1} + f_i$ Для вычислении цепочки из N > 2 чисел Фибоначчи зададим функцию:

```
function f=fib (N)
    f = [1 1] ;
    for 1 = l :N-2
            f ( i + 2 ) = f ( i + l ) + f ( i ) ;
     end
end
```

Теперь, записав функцию под именем fib в директорию WORK, можно вычислить цепочку чисел Фибоначчи от первого числа до N-го (N целое число, большее 2):

```
>> f i b (10)
       ans = 1 1 2 3 5 8 13 21 34 55
```

Приведенная форма файла-функции характерна для функции с одним выходным параметром. Если выходных параметров больше, то они указываются в квадратных скобках после слова function. При этом структура модуля имеет следующий вид:

```
function [vari,var2,...]=f_name(Список_параметров)
%Основной комментарий
%Дополнительный комментарий
Тело файла с любыми выражениями
var1=выражение
var2=выражение
```

Такая функция во многом напоминает процедуру. Её нельзя слепо использовать непосредственно в математических выражениях, поскольку она возвращает не единственный результат, а множество результатов — по числу выходных параметров. Если функция используется, как имеющая единственный выходной параметр, но имеет ряд выходных параметров, то для возврата значения будет использоваться первый из них. Это, зачастую, ведет к ошибкам в математических вычислениях. Поэтому, как отмечалось, данная функция используется как отдельный элемент программ вида:

```
[ v a r l , v a 2 , . . . ] =f_name (Список_параметров)
```

После его применения переменные выхода var1, var2,... становятся определенными и их можно использовать в последующих математических выражениях и иных сегментах программы. Если функция используется в виде f_name (Список_параметров), то возвращается значение только первого выходного параметра — переменной var1.

Переменные, указанные в списке параметров функции, являются *локальными* и служат для переноса значений, которые подставляются на их место при вызовах функций. Возврат из функции производится после обработки всего тела функции, то есть при достижении конца файла функции. При использовании в теле функции условных операторов, циклов или переключателей иногда возникает необходимость осуществить возврат функции раньше, чем будет достигнут конец файла. Для этого служит команда return. В любом случае результатом, возвращаемым функцией, являются значения выходных параметров (в нашем случае выходным параметром является переменная z), присвоенные им на момент возврата.

Итак, из сказанного ясно, что переменные в файлах-сценариях являются *глобальными,* а в файлах-функциях — *локальными.* Нередко применение глобальных переменных в программных модулях может приводить к побочным эффектам. Применение локальных переменных устраняет эту возможность и отвечает требованиям структурного программирования. При необходимости переменные можно сделать глобальными, записав их имена после оператора глобализации global:

```
global var1 var2 …
```

Начиная с версии 5.0 в функции системы MATLAB можно включать *подфункции.* Они объявляются и записываются в теле основных функций и имеют идентичную им конструкцию. Не следует путать эти функции с внутренними функциями, встроенными в ядро системы MATLAB. Подфункции

определены и действуют локально, то есть только в пределах m-файла, определяющего основную функцию. Команда `help name` выводит комментарий, относящийся только к основной функции, тогда как команда `type name` выводит весь листинг m-файла. Так, что заданные в некотором m-файле подфункции нельзя использовать ни в командном режиме работы, ни в других m-файлах.

4. Основы работы с редактором файлов

Для создания m-файлов сценариев и функций в MATLAB имеется специальный редактор/отладчик таких файлов. Его пустое окно открывается командой `New` (Новый файл), которую можно ввести активизацией кнопки с тем же названием в панели инструментов или из позиции `File` меню окна MATLAB. Эта команда выводит подменю со списком типов файлов. Мы будем рассматривать m-файлы.

К примеру, введем такой скрипт-файл:

```
x=0:0.1:15;
y=sin (x);
plot(x,y)
```

Пример ввода листинга этого файла в окне редактора/отладчика m-файла показан на рис. 1.1. Введенный файл можно пустить из окна редактора, исполнив команду Run в позиции Debug (Отладка) меню окна редактора. При этом редактор откроет обычное окно записи файла с заданным именем (например, demol). Рекомендуется записать его в директорию work системы MATLAB. Эта директория будет установлена и в панели инструментов окна системы. При пуске будет вычислено выражение 2+3 и число 5 появится в окне сессии MATLAB, а в отдельном графическом окне будет построен график синусоидальной функции. Все это и видно на рис. 1.1.

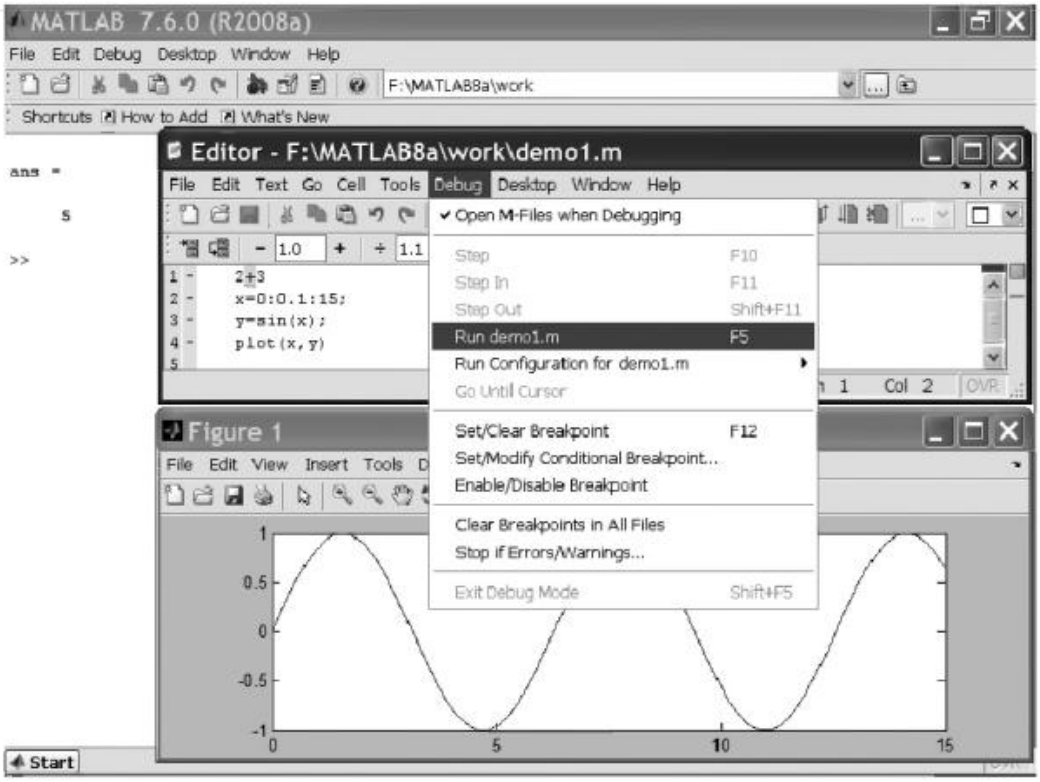

Рис 1.1 пример задания m-файла построения графика синусоиды

Редактор/отладчик m-файлов это в сущности специализированный текстовый редактор, предназначенный для записи и отладки программ на языках системы MATLAB, отдельных их фрагментов, процедур и функций. Строки листинга нумеруются и в них можно вставлять специальные точки останова для отладки сложных программ. В этих точках можно контролировать и изменять значения переменных.

2. Численные методы решения не линейных уравнений

Пусть задана функция $f(x)$. Необходимо найти корни уравнения $f(x) = 0$ Вычислить с заданной точностью ε корень уравнения x , принадлежащий отрезку $[a,b]$. Предполагается, что на отрезке $[a,b]$ находится единственный корень, причем функция $f(x)$ на данном отрезке непрерывна.

2.1 Метод половинного деления

Разделим отрезок [a,b] пополам точкой $C = \frac{a+b}{2}$. Если $f(c) \neq 0$, то возможны два случая

- функция $f(x)$ меняет знак на отрезке $[a, c]$;
- функция $f(x)$ меняет знак на отрезке $[c, b]$.

Выбирая в каждом случае тот отрезок, на котором функция меняет знак, и продолжаем процесс половинного деления до тех пор, пока не выполнится условие

$b - a \leq \varepsilon$

Рисунок 1. иллюстрирует алгоритм нахождение корня методом половинного.

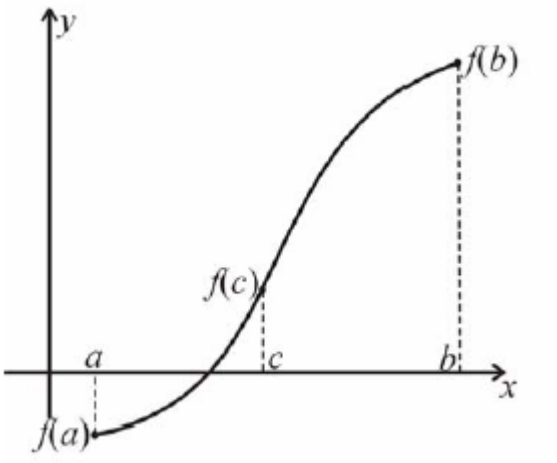

Рис.2.1 Приближение к корню уравнения методом половинного деления

2.2 Метод секущих

Фиксируется некоторая точка x0 ∈[a,b], в которой f (x0)· f "(x0) > 0. Начальным приближением корня уравнения выбирается некоторая точка $x_1 \in [a,b]$ удовлетворяющая условию f (x0)· f (x1) < 0. Итерационная формула метода секущих имеет вид:

$$x_{i+1} = x_{i-1} - \frac{f(x_{i-1}) \cdot (x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$$

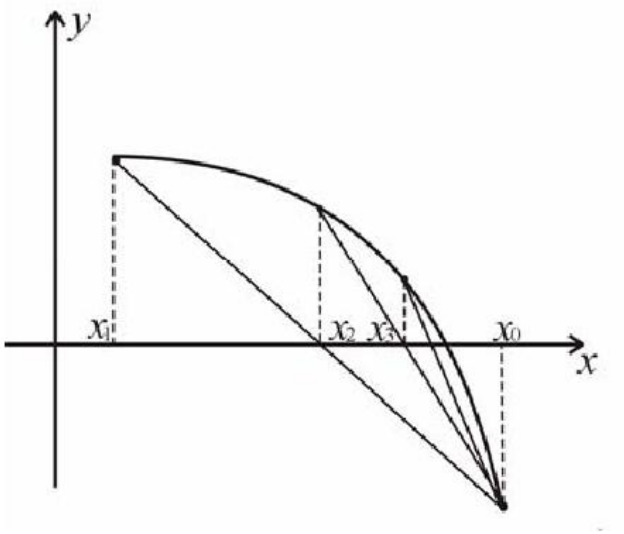

Рис. 2.2 Приближение к корню уравнения методом секущих

Повторять операцию следует до тех пор, $|x_i - x_{i-1}|$ пока не станет меньше или равно заданному значению погрешности ε

Задание 1.

Найти корень уравнения $f(x) = 0$ на отрезке $[a,b]$ с точностью $\varepsilon = 10^{-4}$ методом половинного деления. Решение оформить в виде отдельной функции в m-файле используя средства программирования MATLAB. Построить график функции и убедиться в правильности решения.

Задание 2.

Найти корень уравнения $f(x) = 0$ на отрезке $[a,b]$ с точностью $\varepsilon = 10^{-4}$ методом хорд. Решение оформить в виде отдельной функции в m-файле используя средства программирования MATLAB. Построить график функции и убедиться в правильности решения.

Номер варианта	Уравнение $f(x)=0$	Отрезок $[a,b]$
1	$x^3+0.3\cdot x^2+1.8\cdot x-15=0$	$[2,3]$
2	$x^3+0.2\cdot x^2+5\cdot x+4=0$	$[-1,0]$
3	$x^3+0.7\cdot x^2+4.7\cdot x-1.5=0$	$[0,2]$
4	$x^3+11\cdot x^2-8\cdot x-25=0$	$[1.5,3]$
5	$x^3+3\cdot x^2+3\cdot x+4=0$	$[-3,-2]$
6	$x^3-2\cdot x-0.4=0$	$[1,2]$
7	$x^3+x^2+3\cdot x-13=0$	$[1,3]$
8	$x^3+x^2+3.8\cdot x-2=0$	$[0,1]$
9	$x^3+3.5x^2+12\cdot x+19=0$	$[-3,-1.5]$
10	$x^3+6x^2-17\cdot x-21=0$	$[2,3.5]$