Математические основы проектирования систем управления/

- Решение систем дифференциальных уравнений

Вспомогательный материал связан с особенностями построения m-файлов, процедур и функций в пакете МАТЛАБ, элементами языка программирования.

1. Особенности создания и оформления M-файлов в MatLAB.

Теория

Создание программы в среде MatLAB осуществляется при помощи либо собственного встроенного (начиная с версии MatLAB 5), либо стороннего текстового редактора, который вызывается автоматически, если он предварительно установлен с помощью команды Preferences меню File. Это может быть, например, редактор Notepade среды Windows. *Окно предварительно установленного редактора появляется на экране, если перед этим вызвана команда M-file из подменю New или выбрано название одного из существующих M-файлов при вызове команды Open M-file из меню File. В первом случае окно текстового редактора будет пустым, во втором — в нем будет содержаться текст вызванного M-файла. В обоих случаях окно текстового редактора готово для ввода нового текста либо корректировки существующего.*

В языке MatLAB имеются программы двух типов: так называемые *Script-файлы (файл - сценарии,* или *управляющие программы)* и *файл - функции (процедуры).* Все программы должны иметь расширение имен файлов .m, т.е. их нельзя различить по типу файла. При помощи Script-файлов оформляются основные программы, управляющие от

начала до конца организацией всего вычислительного процесса, и отдельные части основных программ (они могут быть записаны в виде отдельных Script-файлов). Как файл-функции оформляются отдельные процедуры и функции (т.е. такие части программы, которые рассчитаны на неоднократное использование Script-файлами или другими процедурами при изменяемых значениях входных параметров и не могут быть выполнены без предварительного задания значений переменных, которые называют *входными).*

Главным внешним отличием текстов этих двух видов файлов является то, что *файл - функции имеют первую строку вида:*

function <ПКВ> = <имя процедуры >(<ПВВ>)

где ПКВ — Перечень Конечных Величин, ПВВ — Перечень Входных Величин. *Script-файлы такой строки не имеют.*

Принципиальное же отличие заключается в совершенно разном восприятии системой имен переменных в этих файлах.

В файл - функциях все имена переменных внутри файла, а также имена переменных, указанные в заголовке (ПКВ и ПВВ), воспринимаются как *локальные,* т.е. все значения этих переменных после завершения работы процедуры исчезают, и область оперативной памяти ПК, которая была отведена под запись значений этих переменных, освобождается для записи в нее значений других переменных.

В Script-файлах все используемые переменные образуют так называемое *рабочее пространство (work space).* Значения и смысл их сохраняются не только на протяжении работы программы, но и в течение всего сеанса работы с системой, а значит, и при переходе от выполнения одного Script-файла к выполнению другого. Таким образом, рабочее

пространство является единым для всех Script-файлов, вызываемых в текущем сеансе работы с системой. Именно благодаря этому длинный Script-файл можно разбить на несколько фрагментов, оформить каждый из них в виде отдельного Script-файла, а в главном Script-файле вместо соответствующего фрагмента записать оператор вызова Script-файла, представляющего этот фрагмент. Таким образом, обеспечивается компактное и наглядное представление даже довольно сложной программы.

За исключением указанных отличий, файлы-функции и Script- файлы оформляются одинаково.

В дальнейшем под М-файлом будем понимать любой файл (файл-функцию или Script-файл), записанный на языке системы MatLAB.

Рассмотрим основные особенности записи **текста** программы (М-файла) на языке MatLAB.

• Обычно каждый оператор записывается в отдельной строке текста программы. Признаком конца оператора является символ (он не появляется в окне) возврата каретки и перехода на следующую строку, который вводится в программу при нажатии клавиши [Enter], т.е. при переходе на следующую строку.

• Можно размещать несколько операторов в одной строке. Тогда предыдущий оператор этой строки должен заканчиваться символом";" или ",".

• Длинный оператор можно записывать в несколько строк**.** При этом предыдущая строка оператора должна заканчиваться тремя точками (...).

• Если очередной оператор не заканчивается символом ";", результат его действия при выполнении программы будет выведен в командное окно. Поэтому для предотвращения вывода на экран результатов действия оператора программы, запись этого оператора в тексте программы должна заканчиваться символом ";".

• Строка программы, начинающаяся с символа "%", не выполняется. Эта строка воспринимается системой MatLAB как *комментарий.* Таким образом, для ввода комментария в любое место текста программы достаточно начать соответствующую строку с символа "%".

• Строки комментария, предшествующие первому выполняемому оператору программы, т.е. такому, который **не** является комментарием, воспринимаются системой MatLAB как описание программы. Именно эти строки выводятся в командное окно, если в нем набрана команда:

help <имя файла>

В программах на языке MatLAB отсутствует оператор окончания текста программы.

В языке MatLAB переменные **не** описываются и н**е** объявляются. Любое новое имя, появляющееся в тексте программы, воспринимается системой MatLAB как имя матрицы. Размер этой матрицы устанавливается при предварительном вводе значений ее элементов либо определяется действиями по установлению значений ее элементов, описанными в предыдущем операторе или процедуре. Эта особенность делает язык MatLAB очень простым в употреблении и привлекательным. В языке MatLAB невозможно использование матрицы или переменной, в которой предварительно не введены или не вычислены значения ее элементов (а значит, и не определены размеры этой матрицы). В этом случае при выполнении программы MatLAB появится сообщение об ошибке "Переменная не определена".

• Имена переменных могут содержать лишь буквы латинского алфавита или цифры и должны начинаться с буквы. Общее число символов в имени может достигать 19. В именах переменных могут использоваться как прописные, так и строчные буквы. Особенностью языка MatLAB является то, что *прописные и строчные буквы в именах различаются системой.* Например, символы "a" и "A" могут использоваться в одной программе для обозначения разных величин.

2. Язык программирования

Описание	Действие	Результат
Общие положения		
Операторы управления вычислительным процессом Вообще, операторы управления необходимы главным образом, для организации вычислительного процесса, который записывается в виде некоторого текста программы на языке программирования высокого уровня. При этом к операторам управления вычислительным процессом обычно относят операторы безусловного перехода, условных переходов (разветвления вычислительного процесса) и операторы организации циклических процессов. Однако система MatLAB построена таким образом, что эти операторы могут быть использованы и при работе MatLAB в режиме калькулятора. В языке *MatLAB отсутствует оператор безусловного перехода, и поэтому нет понятия метки.* Это является недостатком языка MatLAB и затрудняет организацию возвращения вычислительного процесса к любому предыдущему или последующему оператору программы. Все операторы цикла и условного перехода построены в MatLAB виде сложного оператора, который начинается служебным словом *if, while, switch* или *for* и заканчивается служебным словом *end*. Операторы между этими словами воспринимаются системой как части одного сложного оператора. Поэтому нажатие клавиши *enter*| для перехода к следующей строке не приводит в данном случае' к выполнению этих операторов. Выполнение операторов начинается только тогда, когда введена		

"закрывающая скобка" сложного оператора в виде слова *end*, а затем нажата клавиша [Enter]. Если несколько сложных операторов такого типа вложены один в другой, вычисления начинаются лишь тогда, когда записан конец (*end*) наиболее охватывающего (внешнего) сложного оператора. Из этого вытекает возможность осуществления даже в режиме калькулятора довольно сложных и объемных (состоящих из многих строк и операторов) вычислений, если они охвачены сложим оператором.		
Оператор условного перехода		
Конструкция оператора перехода по условию в общем виде такова: *if <условие>* *<операторы1>* *else* *<операторы2>* *end* Работает он следующим образом. Вначале проверяется, выполняется ли указанное условие. Если да, то программа выполняет совокупность операторов, которая записана в разделе <операторы1>. В противном случае выполняется последовательность операторов раздела <операторы2>. Укороченная форма условного оператора имеет вид: *if <условие>* *<операторы>* *end* Действие оператора в этом случае аналогично, за исключением того, что при невыполнении заданного условия выполняется оператор, следующий за оператором end. Легко заметить недостатки этого оператора, вытекающие из		

отсутствия оператора безусловного перехода: вся часть программы, выполняющаяся в зависимости от условия, должна размещав внутри операторных скобок *if* и *end*. В качестве условия используется выражение типа: <имя переменной1> <операция сравнения> <имя переменной2> *Операции сравнения* в языке MatLAB могут быть такими: < меньше > больше <= меньше или равно >= больше или равно == равно ~= не равно Условие может быть составным, т.е. складываться из нескольких. И» простых условий, объединяемых знаками логических операций, какими логических операций в языке MatLAB являются: & логическая операция И (AND) | логическая операция ИЛИ (OR) ~ логическая операция НЕТ (NOT) Логическая операция Исключающее ИЛИ может быть реализована при помощи функции *хог(А,В,* где А и В — некоторые условия. Допустима еще одна конструкция оператора условного перехода: if <условие1> <операторы> elseif <условие2> <операторы2> elseif <условие3> <операторы3> ….		

else <операторы> Оператор elseif выполняется тогда, когда <условие1> не выполняется. При этом сначала проверяется <условие2>. Если оно выполнено, выполняются <операторы2>, если же нет, то <операторы2> игнорируются и происходит переход к следующему оператору elseif, т.е. к проверке <условия3>. Аналогичным образом при его выполнении обрабатываются <операторы3>, в противном случае происходит переход к следующему оператору elseif. Если ни одно из условий в операторах elseif не выполнено, обрабатываются <операторы>, следующие за оператором else. Таким образом, может быть обеспечено ветвление программы по нескольким направлениям.		
Оператор переключения		
Оператор переключения имеет такую структуру: *switch <выражение, скаляр или строка символов>* *case <значение1>* *<операторы1>* *case <значение2>* *<:операторы2>* … *othewise* *<операторы>* *end* Он осуществляет ветвление вычислений в зависимости от значений некоторой переменной или выражения, сравнивая значение, полученное в результате вычисления выражения в строке *switch*, со значениями, указанными в строках со словом *case*. Соответствующая группа операторов *case* выполняется, если значение выражении совпадает со значением,		

указанным в соответствующей строке *case*. Если значение выражения не совпадает ни с одним из значений и группах *case*, выполняются операторы, следующие за *otherwise*.	
Операторы цикла	
В языке MatLAB есть две разновидности операторов цикла: *условный* и *арифметический*. *Оператор цикла с предусловием имеет* вид: *while <условие>* *<операторы>* *end* Операторы внутри цикла обрабатываются лишь в том случае, если выполнено условие, записанное после слова **while**. При этом среди операторов внутри цикла обязательно должны быть такие, которые изменяют значения одной из переменных, указанных в условии цикла. *Арифметический оператор цикла* имеет вид: for <имя> = <НЗ> : <Ш> : <КЗ> <операторы> end где <имя> — имя управляющей переменной цикла — счетчика цикла; <НЗ> — заданное начальное значение этой переменной; <Ш> — значение	

шага, с которым она должна изменяться; <КЗ> — конечное значение переменной цикла. В этом случае <операторы> внутри цикла выполняются несколько раз (каждый раз при новом значении управляющей переменной) до тех пор, пока значение управляющей переменной не выйдет за пределы интервала между <НЗ> и <КЗ>. Если параметр <Ш> не указан, по умолчанию его значение принимается равным единице. Чтобы досрочно выйти из цикла (например, при выполнении некоторого условия), применяют оператор break. Если в программе встречается этот оператор, выполнение цикла досрочно прекращается и начинает выполняться следующий после слова end оператор.	

3. Решение системы дифференциальных уравнений в форме Коши средствами МАТЛАБ

Теория

Интегрирование обыкновенных дифференциальных уравнений осуществляют функции **ode23 и ode45.** Они могут применяться как для решения простых дифференциальных уравнений, так и для моделирования сложных динамических систем.

Известно, что любая система обыкновенных дифференциальных уравнений (ОДУ) может быть представлена в так называемой форме Коши:

$$\frac{dy}{dt} = f(y,t)$$

где *y* — вектор переменных состояния системы; t— аргумент (обычно время); f— нелинейная вектор-функция от переменных состояния *y* и аргумента t.

Обращение к процедурам численного интегрирования ОДУ имеет вид:

[t, y] = ode23 ('<имя функции>', tspan, y0, options)

[t, y] = ode45 ('<имя функции>', tspan, y0, options)

где <имя функции> — строка символов, являющаяся **именем** М-файла, в котором вычисляется вектор-функция *f(y,t),* т.е. *правые части системы ОДУ;*

y0 — вектор начальных значений переменных состояния;

t — массив значений аргумента, соответствующих шагам интегрирования;

y — матрица проинтегрированных значений фазовых переменных, в которой каждый столбец соответствует одной из переменных состояния, а строка содержит значения переменных состояния, соответствующих определенному шагу интегрирования;

tspan — вектор-строка [t0 tfinal], содержащая два значения: t0 — начальное значение аргумента t; tfinal — конечное значение аргумента;

options — строка параметров, определяющих значения допустимой относительной и абсолютной погрешности интегрирования.

Параметр options можно не указывать. Тогда по умолчанию допустимая относительная погрешность интегрирования принимается равной 1.0e-3, а абсолютная (по каждой из переменных состояния**)** — 1.0e-6**.** Если же эти значения не устраивают пользователя, следует перед обращением к процедуре численного интегрирования установить новые значения допустимых погрешностей и с помощью процедуры odeset таким образом:

options= odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-5])

Параметр RelTol определяет относительную погрешность численного интегрирования по всем переменным одновременно, а AbsTol является вектором-строкой, состоящим из абсолютных допустимых погрешностей численного интегрирования по каждой из фазовых переменных.

Функция **ode23** осуществляет интегрирование численным методом Рунге-Кутта 2-го порядка, а с помощью метода.3-го порядка контролирует относительные и абсолютные ошибки интегрирования на каждом шаге и изменяет величину шага интегрирования так, чтобы обеспечить заданные пределы ошибок интегрирования. Для функции **ode45**

основным методом интегрирования является метод Рунге-Кутта 4-го порядка, а величина шага контролируется методом 5-го порядка.

Практика

Исследовать модель средствами МАТЛАБ при представлении в виде системы дифференциальных уравнений в форме Коши

Порядок действий

1. Взять систему в нормальной форме Коши

$$\frac{dq_д}{dt} = f_1(q_д, \omega_д, i_я, ...)$$

$$\frac{d\omega_д}{dt} = f_2(q_д, \omega_д, i_я, ...)$$

$$\frac{di_я}{dt} = f_3(q_д, \omega_д, i_я, ...)$$

2. Создать m-файл под именем myparam.m для ввода параметров системы с текстом (рис.1)

```
km=0.36;   % Km
kv=0.45;   % Kc
R=0.5;
L=0.01;
Jd=0.04;    % Jд
k2=1/Jd;    % (1/Jд)
k1=1/R;
T=L/R;
```

Рис.1

3. Создать m-файл для расчета правых частей системы дифференциальных уравнений под именем myfun.m со следующим текстом (рис.2)

```
function dydt=myfun(t,y)
% расчет вектора производных от вектора y переменных состояния
global Mv u km kv  Jd L R;
q=y(1);
wd=y(2);
ia=y(3);
dydt(1)=wd;
dydt(2)=(km*ia-Mv)/Jd;
dydt(3)=((u-kv*wd)-R*ia)/L;
dydt=dydt';
```

Рис.2

4. Сформировать организующий m- файл, в котором будет текст (рис.3)

```
% описание глобальных параметров
global Mv u km kv  Jd L R;
 u=1; Mv=1;
% ввод начальных данных
y0(1)=0; % q
y0(2)=0; % wd
y0(3)=0; % ia
% вызов m- файла с параметрами задачи
myparam
% задание tspan - вектор-строка [t0 tfinal], содержащаЯ два значения:
% t0 - начальное значение аргумента t; tfinal - конечное значение аргумента;
t0=0;
tfinal=1;
tspan=[t0 tfinal];
% задание опций
options= odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-5]);
% вызов программы решениЯ
[t, y] = ode45 ('myfun', tspan, y0, options);
% построение графика в 3 подокнах
subplot(3,1,1); plot(t,y(:,1)); xlabel('t'); ylabel('q')
subplot(3,1,2); plot(t,y(:,2)); xlabel('t'); ylabel('q''')
subplot(3,1,3); plot(t,y(:,3)); xlabel('t'); ylabel('i')
```

Рис.3

5. Запустить на выполнение программу, набрав в командной строке МАТЛАБ имя организующего m- файла
6. Сравнить графики (рис.4) с результатами моделирования в других представлениях системы.