Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина
Уральский энергетический институт

Г. Б. СМИРНОВ
В. Г. ТОМАШЕВИЧ

ЛИНЕЙНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ В ПАКЕТЕ MATLAB

Учебное пособие

Министерство образования и науки Российской Федерации
Уральский федеральный университет
имени первого Президента России Б. Н. Ельцина

Г. Б. Смирнов
В. Г. Томашевич

ЛИНЕЙНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ В ПАКЕТЕ MATLAB

Учебное пособие

Рекомендовано методическим советом
Уральского федерального университета
для студентов вуза, обучающихся
по направлению подготовки
13.03.02 — Электроэнергетика и электротехника

Екатеринбург
Издательство Уральского университета
2018

УДК 681.511.2:004.4(075.8)
ББК 32.972я73+32.965.4я73
С50

Авторы:
Г. Б. Смирнов, В. Г. Томашевич

Рецензенты:
доц., канд. физ.-мат. наук А. В. Кибардин (кафедра «Информационные технологии и защита информации» УрГУПС);
завкафедрой «Электроэнергетика» УГЛТУ проф., д-р техн. наук С. М. Шанчуров)

Научный редактор — проф., д-р техн. наук В. Э. Фризен

Смирнов, Г. Б.
С50 Линейные системы управления в пакете MATLAB : учебное пособие / Г. Б. Смирнов, В. Г. Томашевич. — Екатеринбург : Изд-во Урал. ун-та, 2018. — 76 с.

ISBN 978-5-7996-2385-2

Учебное пособие предназначено для первоначального знакомства с современным интерактивным пакетом MATLAB, применяемым в инженерном деле, в математике и экономике. В него включены элементарные сведения по работе с векторами и матрицами, рассмотрены арифметические операции и функции. Приводятся сведения, позволяющие использовать богатые возможности пакета по графическому представлению данных. Рассмотрены примеры моделирования и проектирования простейших регуляторов с использованием MATLAB/SIMULINK. В пособие включены также основные принципы синтеза регуляторов систем подчинённого регулирования и приведены данные по типовым переходным функциям этих систем. Представлены основные команды, позволяющие работать с матрицами и структурами, которые представляют объекты из области теории управления техническими системами. Пособие предназначено для студентов всех форм обучения по направлению подготовки 13.03.02 — Электроэнергетика и электротехника.

Библиогр.: 11 назв. Рис. 7.

УДК 681.511.2:004.4(075.8)
ББК 32.972я73+32.965.4я73

ISBN 978-5-7996-2385-2

Введение

У истоков MATLAB стоял Клив Молер (Cleve Moler), работавший в 1970-х гг. в университете Нью Мехико (New Mexico). Сначала он хотел обеспечить своим студентам комфортный доступ к библиотекам линейной алгебры Linpack и Eispack, написанным на языке программирования FORTRAN таким образом, чтобы для этого не требовалось серьёзных знаний в программировании. В дальнейшем, в 1984 г. Клив Молер вместе с Джеком Литтлом (Jack Little) и Стивом Бангертом (Steve Bangert) основал в Натике (Natick, Massachusetts, USA) фирму The Mathworks, которая превратила MATLAB в коммерческий продукт и стала развивать его дальше.

За прошедшее с тех пор время MATLAB превратился в универсальный инструмент инженера и учёного. Сегодня это своеобразный язык программирования, предназначенный для решения математических, физических и научно-технических задач. Сегодня MATLAB — это язык программирования четвертого поколения, главной особенностью которого является возможность быстро составлять эффективные прикладные программы. Он оптимизирован для работы с матрицами и выполнения численных расчётов и своё название берёт от слов Matrix Laboratory.

SIMULINK — расширение MATLAB, позволяющее создавать модели, базирующиеся на использовании дифференциальных уравнений и графических блоков, как это бывает, например, в теории систем, теории управления и теории обработки сигналов [1, 2, 4, 5].

Сегодня MATLAB/SIMULINK — это интерактивный пакет для вычислений в инженерной практике, по факту являющийся международным стандартом для моделирования технических систем не только в высшей школе, но и в промышленности.

Объём функций пакета можно расширить благодаря применению добавочных пакетов Toolbox (MATLAB) и Blockset (SIMULINK), при-

чём эти функции и блоки используются для определённых научных дисциплин.

В качестве примера можно упомянуть:

- пакет идентификации систем;
- пакет по обработке сигналов и изображений;
- пакет по вейвлетам;
- пакет по финансово-экономическим расчетам;
- пакет для построения нейронных сетей;
- пакет, относящийся к теории размытых множеств;
- пакет SimPowerSystem по моделированию в электроэнергетике;
- пакет SimMechanics по моделированию в механике
- и т. д.

Некоторые пакеты оказались настолько интегрированными с системой MATLAB, что стали её составной частью. Это относится к уже упомянутому пакету SIMULINK (управление) и Notebook (интеграция с текстовым процессором, что позволяет создавать электронные документы и книги с примерами математических расчетов и высокой степенью графической визуализации всех этапов решения задачи).

MATLAB предоставляет следующие возможности:

- интерактивная работа с помощью интерпретирующего языка через командное окно Command Window;
- альтернативное использование M-файлов, содержащих команды MATLAB;
- использование в моделировании так называемых Toolbox-ов, которые являются готовыми M-файлами и могут использоваться как дополнение к обычным командам MATLAB, предлагая наборы команд в специальных областях. Специальным Toolbox-ом является, например, средство моделирования SIMULINK, с помощью которого могут быть составлены модели из готовых графических блоков. Примером является также инструмент Control Toolbox, команды которого используются при моделировании в области атоматического управления. Другим примером является средство Signal Processing Toolbox, которое используется для обработки сигналов.

Особенное значение в MATLAB имеет работа с матрицами. Отсюда вытекают некоторые особенности при работе в пакете.

Важным достоинством системы MATLAB является ее открытость и расширяемость. Большинство команд и функций данной системы

оформлены в виде текстовых файлов (M-файлов) и файлов на языке C (C++). Пользователь может их модифицировать и создавать новые. Также в MATLAB есть возможность объединения системы с пакетом символьной математики Maple, пакетом Excel и некоторыми другими.

Области применения пакета:

- математические вычисления;
- разработка алгоритмов;
- моделирование;
- накопление, анализ и обработка данных: оценка и визуализация результатов;
- вычислительный эксперимент, имитационное моделирование, макетирование;
- использование графики в математике и технике;
- разработка приложений, в том числе с графическим интерфейсом [1–3].

Данное учебное пособие адаптировано для решения задач управления в электротехнике.

Примечание. Фирма The Mathworks предлагает студенческую версию своего продукта (MATLAB): (http://www.mathworks.com).

Глава 1.
Основы работы с MATLAB

1.1. Интерактивная работа в командном окне пакета MATLAB Command Window

Запуск системы осуществляется стандартным для Windows способом (двойной щелчок левой клавишей мыши на ярлыке MATLAB в рабочем меню операционной системы). После запуска программы MATLAB на дисплее компьютера появляется ее главное окно (рис. 1.1), и система готова к проведению вычислений в командном режиме.

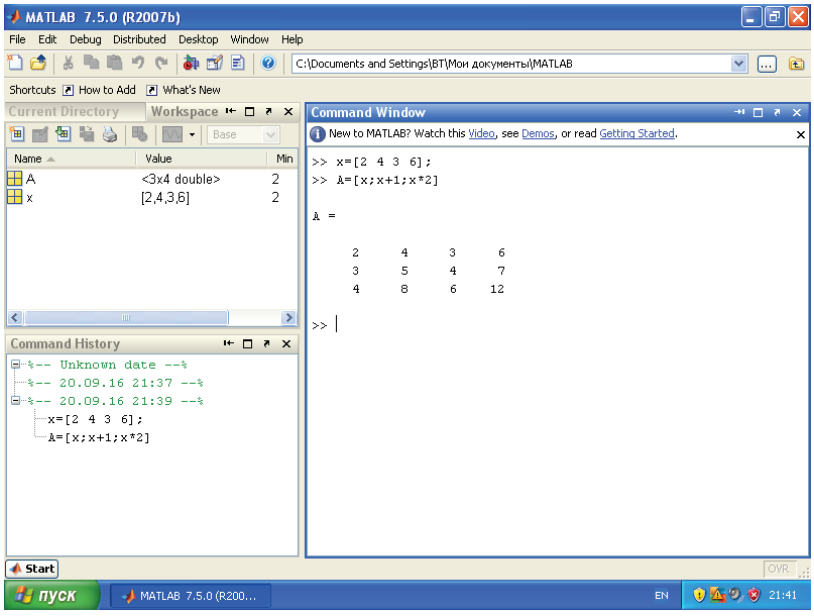


Рис. 1.1. Командное окно MATLAB

Все команды и определения переменных задаются в командном окне MATLAB после символов >>. Обычно командное окно занимает правую половину экрана. Если это не так, то стандартное представление может быть восстановлено следующей последовательностью команд: **Desktop** → **Desktop Layout** → **Default**.

1.2. Описание команды help MATLAB

Пункт меню **help** позволяет воспользоваться подробной справочной системой и демонстрационными примерами. На эти примеры можно также выйти, набрав в командном окне команду **demo**. В демонстрационных примерах содержится большое число серьезных задач. При необходимости можно ознакомиться с файлами примеров и даже перенести тексты программ в командное окно MATLAB, используя буфер промежуточного хранения.

Чтобы получить справку по какому-либо объекту, необходимо в командном окне набрать: **help <имя объекта>**.

Если нет уверенности в правильном написании имени объекта или оно неизвестно, можно использовать поиск по ключевому слову или по последовательности слов.

Команда **help** осуществляет вызов всех каталогов **help**, которые имеются в MATLAB, в числе которых, например, может находиться каталог **matfun**, где находятся функции линейной алгебры [2–4]. Можно распечатать все функции этого каталога, если использовать команду **help matfun**. Соответственно, **help elfun** распечатает список элементарных функций и т. д. Команда **help log** — это уже вызов помощи по определённой функции. Для получения справки можно использовать также команду **lookfor**; есть возможность использовать окно **help** рабочего стола MATLAB и т. д.

Примечание. Поиск помощи по команде возможен, если команда известна, в противном случае возникают затруднения. Для облегчения поиска важнейшие команды MATLAB и Toolbox-ов из области теории управления представлены в Приложении 2. С помощью команд из этого списка можно решить большинство задач, встречающихся в практике обучения высшей школы по курсу «Основы теории управления».

Глава 2.
Числа. Векторы и матрицы

Основополагающая структура в MATLAB — комплексная матрица [1–3]. Все другие структуры, такие, как векторы и скаляры, — частные случаи матрицы. Некоторые параметры специальных функций также представлены векторами. Измеряемые величины обрабатываются и хранятся как векторы или матрицы. Матрицы в MATLAB задаются определённым образом.

2.1. Форматы чисел

Сначала необходимо сказать об основных формах представления чисел в MATLAB. Возможны следующие представления: 4; –34; 0.0012; 8.5; 67; –4; 1.56E-1; 14.5e8. Из приведённых примеров становятся ясны основные правила записи чисел. Следует обратить внимание на то, что при задании чисел в экспоненциальном формате нельзя ставить пробелы перед E(e). Неверной была бы, например, запись 5.6 E-7. Порядок чисел в MATLAB от 10E-308 до 10E+308. При использовании комплексных чисел возможно применение в качестве мнимой единицы i или j: (5+3*i–7*j). В пакете используются различные форматы при работе с числами, т. е. числа на экране могут быть представлены различным числом значащих цифр. Формат может быть изменен с помощью специальных команд, но стандартным форматом является **short**. Следующие примеры показывают правила применения форматов:

Формат (команда)	Представление числа 4/3	Представл. числа 1.2345e-6
format short	1.3333	0.0000
format short e	1.3333E+000	1.2345E-006
format long	1.3333333333333338	0.000001234500000
format long e	1.3333333333333338E+000	1.234500000000000E-006
format hex	3FF555555555555555	3EB4B6231ABFD271

Следует сказать, что сами вычисления в пакете выполняются с двойной точностью.

2.2. Определение переменных как скаляров, векторов или матриц

При задании величин в виде векторов или матриц следует иметь в виду, что эти величины должны быть взяты в квадратные скобки. Число пи задано константой pi.

Задание простых величин: скаляров, векторов, матриц

`>>8` `ans=8`	При задании числа и последующем нажатии клавиши Enter MATLAB присвоит величину временной переменной ans (сокращение от английского *answer*)
`>>a=8` `a=8`	Обычно MATLAB повторяет содержание переменной в окне
`>>a=8;`	Повторная выдача переменной в окне подавляется, если после значения переменной поставить точку с запятой

`>>a=8, b=4`
`a=8`
`b=4`
	При разделении переменных запятой они повторяются в следующих строках, если переменные не заканчиваются точкой с запятой

`>>a=8; b=3;`	Если заданные в одной строке переменные разделяются точкой с запятой, то они повторно не выводятся

Задание векторов-столбцов

`>>x=[3;4;5];`	Завершение вектора-столбца точкой с запятой подавляет повторный вывод вектора

`>>x=[3;4;5]`
`x=3`
`4`
`5`
	При отсутствии точки с запятой после угловых скобок вектор-столбец будет повторно выдан на экран. Иногда это хорошо для контроля, но плохо, если вектор-столбец очень длинный

Задание векторов-строк

`>>y=[1 2 3 4 5];`	Отдельные элементы вектора-строки могут разделяться пробелами или запятыми
`>>y=[1,2,3,4,5];`	Заключительная точка с запятой подавляет повторную выдачу содержания переменных

`>>y=3:6` `y=3 4 5 6`	Этот оператор задаёт вектор-строку со значениями «от и до» с шагом 1
`>>y=1:2:7` `y=1 3 5 7`	Такой оператор задаёт вектор и его элементы с шагом 2 от 1 до 7. Подобным образом можно задавать векторы с произвольными начальными и конечными значениями и шагом

Определение значений элементов матриц

`>>A=[1 2 3;4 5 6];`	Элементы матрицы разделяются пробелами, строки — точкой с запятой

`>>A=[1 2 3;4 5 6]`
`A=1 2 3`
`4 5 6`
	Матрица распечатывается, если не ставится точка с запятой после закрывающей квадратной скобки

`>>A=[1 2 3`
`4 5 6]`
`A=1 2 3`
`4 5 6`
	Альтернативная форма определения матрицы: в этом случае строки должны заканчиваться нажатием клавиши Enter

`A=[1 2 3*4`
`10/2 3+1 8]`
`A=1 2 1 2`
`5 4 8`
	Элементы матрицы могут также задаваться как результат вычислений

`>>x=[1;3;5];`
`>>y=[2;4;6];`
`>>A=[x y]`
`A=1 2`
`3 4`
`5 6`
	Если x и y матрицы — столбцы равной длины, то можно составить матрицу A, состоящую из столбцов x и y

`>>x=[0 1 2 3 4];`
`>>A=[x; x-1; x*2]`
`A= 0 1 2 3 4`
`-1 0 1 2 3`
`0 2 4 6 8`
	Из нескольких векторов-строк с использованием точки с запятой и арифметических операций можно составить матрицу

Извлечение элементов, строк и столбцов из матрицы

`>>x=A(2,3)` `x=1`	Пример извлечения элемента матрицы A из строки с номером 2 и столбца с номером 3

`>> x=A(:,2)`
`x=1`
`0`
`2`
	Пример формирования вектора-столбца из столбца с номером 2 матрицы A

`>>x=A(1:3,2)`
`x=1`
`0`
`2`
	Пример формирования вектора-столбца из элементов столбца с номером 2 и элементов строк с номерами 1, 2, 3

`>> x=A(3,:)` `x=02468`	Пример формирования вектора-строки из строки матрицы A с номером 3
`>>x=A(2,2:5)` `x = 0123`	Пример формирования вектора-строки из элементов строки номер 2 и столбцов с номерами 2, 3, 4, 5

Примечание. Возможность извлечения векторов-столбцов и векторов-строк из матрицы очень полезна при частичном использовании данных, хранящихся в виде матриц и нуждающихся в особой обработке или графическом представлении.

Определение нулевой матрицы

`>>y=zeros(2,4)`
`y=0000`
`0000`
	Пример формирования нулевой матрицы с числом строк 2 и числом столбцов 4

Определение единичной матрицы

`>>E=eye(3)`
`E=100`
`010`
`001`
	Единичная матрица размерностью 3×3 формируется по команде **`eye`**

Просмотр всех ранее созданных переменных

`>>who`
`our variables are:`
`A a ans b c x y`
	Все переменные, хранящиеся на поверхности (**`Workspace`**) в MATLAB на данный момент времени, будут распечатаны

`>>whos`	Эта команда позволяет более точно узнать вид, длину и способ представления переменных (см. рис. 2.1)

Примечание. В MATLAB имеет значение регистр при назначении имён переменным (aB и AB — разные идентификаторы). Многие сообщения об ошибках — результат некорректного использования регистра, поэтому необходимо выбирать простые, логичные и возможно короткие имена.

```
Name        Size                Bytes  Class     Attributes

A           5x3                   120  double
E           3x3                    72  double
a           1x1                     8  double
b           1x1                     8  double
x           4x1                    32  double
y           2x4                    64  double
```

Рис. 2.1. Описание переменных MATLAB, распечатанных в командном окне по команде **whos**

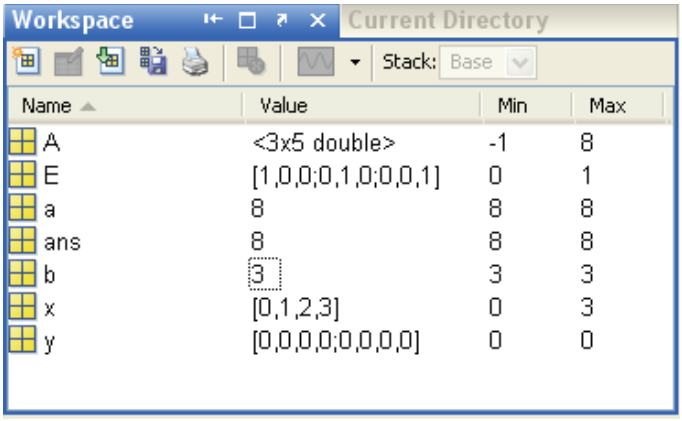


Рис. 2.2. Описание переменных MATLAB на панели **Workspace**

Сохранение переменных

Переменные, которые находятся на панели **Workspace**, хранятся только один сеанс. После окончания сеанса все созданные переменные теряются. Переменные, которые необходимо сохранить, можно запомнить с помощью команды **save**.

>>save A	Без указания имени матрица сохранится как mat-файл в форме A.mat в текущем каталоге MATLAB
>>save Name A E a ans b x y	Если необходимо сохранить несколько переменных, то их можно разделить пробелами и сохранить в mat-файле с заданным именем (Name)

>>save Name	При указании после команды **save** имени mat-файла все переменные рабочего стола будут сохранены. Все переменные рабочего стола можно сохранить также с помощью последовательности **File** → **Save Workspace As**... Имя mat-файла не может быть использовано в качестве имени переменной!

Загрузка сохранённых переменных

>>load name	Сохранённую при помощи команды **save** переменную можно снова загрузить на рабочую поверхность. В этом случае при задании имени переменной регистр не играет никакой роли (можно указать имя Name или NAmE)

Глава 3.
Простейшие арифметические операции и функции

3.1. Основные арифметические операции для скалярных величин (матриц размерностью 1x1)

>>c=a+b — сложение (+).
>>c=a-b — вычитание (-).
>>c=a*b — умножение (*).
>>c=a/b — деление (/).
>>c=a^b — возведение в степень (^).

3.2. Тригонометрические функции

>>sin(x)	**>>asin(x)**	**>>sinh(x)**	**>>asinh(x)**
>>cos(x)	**>>acos(x)**	**>>cosh(x)**	**>>acosh(x)**
>>tan(x)	**>>atan(x)**	**>>tanh(x)**	**>>atanh(x)**

При записи функций применяют круглые скобки, а при задании матриц или векторов — квадратные.

3.3. Элементарные функции

>>abs(x) — абсолютная величина x.
>>angle(x) — аргумент числа.
>>sign(x) — знаковая функция.

>>real(x) — действительная часть.
>>sqrt(x) — квадратный корень.
>>imag(x) — мнимая часть.
>>exp(x) — экспонента.
>>conj(x) — число, сопряжённое x.
>>log(x) — логарифм натуральный.
>>round(x) — округление до целого.
>>log10(x) — логарифм десятичный.
>>rem(a,b) — остаток после деления.

3.4. Операции отношений

Здесь рассматриваются операции отношения, результат которых или 0, или 1. Эти операции применимы также к матрицам и векторам соответствующих размерностей. Результатом будет вектор или матрица.

>> x < y — меньше.
>> x == y — равно.
>> x > y — больше.
>> x <= y — меньше или равно.
>> x ~= y — не равно.
>> x >= y — больше или равно.

3.5. Векторы и матрицы: основы работы

При работе с векторами и матрицами необходимо согласовывать размерности, придерживаясь соответствующих правил.

Матричные операции

С приведёнными ниже матричными операциями можно познакомиться или в разделе **help** MATLAB или в Приложении 1.

>>u=[1 2 3]; v=[4 5 6]; w=[8 9 0; 4 5 6; 1 2 3];	
>>u+v ans = 5 7 9 >>u-v ans = -3-3-3	Сложение/вычитание осуществляется поэлементно
>>u+[3 4] ??? Error using -> + Matrix dimensions must agree	Сообщение об ошибке, т. к. была попытка сложить векторы разной размерности
>>t=w′ t=8 4 1 9 5 2 0 6 3	Операция транспонирования матрицы. Если w — комплексная матрица, то операция транспонирования даёт комплексно-сопряжённую матрицу
>>inv(w)	Операция получения обратной матрицы
>>rank(w)	Определение ранга матрицы (возвращает количество линейно-независимых строк (столбцов))
>>det(w)	Возвращает значение определителя (детерминанта) квадратной матрицы
>>w*t ans = 145 77 26 77 77 32 26 32 14	Умножение матриц
>>u*v ??? Error using ->* Matrix dimensions must agree	Сообщение об ошибке, т. к. была попытка перемножить векторы-строки
>>u′*v ans = 4 5 6 8 10 12 12 15 18	Перемножение вектор-столбца на вектор-строку
>>u*v′ ans = 32	Перемножение вектора-строки на вектор-столбец. Результат — скаляр
>>u*3 ans = 3 6 9	Умножение вектора-строки на скаляр
>>w^2 ans = 100 117 54 58 73 48 19 25 21	Пример возведения матрицы в степень 2
>> u*w ans = 19 25 21	Пример умножения вектора-строки на матрицу
>>w*u′ ans = 26 32 14	Пример умножения матрицы на вектор-столбец

`>>x=w\t`
`x = 7.300 -6.700 -3.700`
`  -5.600  6.400  3.400`
`  1.300 -0.033 -0.033`
	Пример решения линейной системы уравнений w*x=t (см. **help mldivide**)

`>>x=w\u'`
`x = -3.7000`
`  3.4000`
`  -0.0333`
	Пример решения линейной системы уравнений w*x=u'. Применяют при использовании алгоритма Гаусса

`>>x=t/w`
`x = -0.1000  4.2333 -8.1333`
`  -0.1000  4.5667 -8.4667`
`  0.9000 -4.1000  9.2000`
	Пример решения матричного уравнения x*w=t (см. **help mrdivide**)

`>>x=eig(w)`
`x =  13.5485`
`  3.1536`
`  -0.7021`
	Вектор x представляет собственные значения матрицы w

Поэлементное умножение и деление двух векторов

`>>u.*v` `ans = 4 10 18`	Поэлементное перемножение элементов двух матриц. Для выполнения операции используется точка (см. **help times**)
`>>x=u.\v` `x=4.000 2.500 2.000`	Поэлементное деление матриц предваряется точкой в команде. Решение уравнения u*x=v (см. **help ldivide**)
`>>x=u./v` `x= 0.250 0.400 0.500`	Решение уравнения v*x = u (см. **help rdivide**)
`>>x=u.^v` `x=1 32 729`	Поэлементное возведение в степень; v должен быть вектором или скаляром
`>>x=u.^3` `x=1 8 27`	Поэлементное возведение в степень; по-прежнему необходимо использовать точку

Примечание. Более подробную информацию об операциях над матрицами можно получить в руководстве по MATLAB — **Getting Started** в разделе **Manipulating Matrices**.

3.6. Построение графиков функций

Простейшие графики

Простейшей командой для выдачи графиков на экран является команда **plot**. Эта команда также использует или вектор, содержащий, например, значения ординаты y, или матрицу, содержащую, как минимум, пары значений координат x, y. Если вектор содержит значения ординат, то MATLAB в качестве абсцисс использует последовательность 1, 2, 3, ... Каждый график при этом будет выдаваться в отдельном окне. Если же друг за другом выдаётся несколько графиков, то все они размещаются в одном окне до тех пор, пока новое окно не будет открыто командой **figure** или текущее окно не будет сохранено командой **hold**.

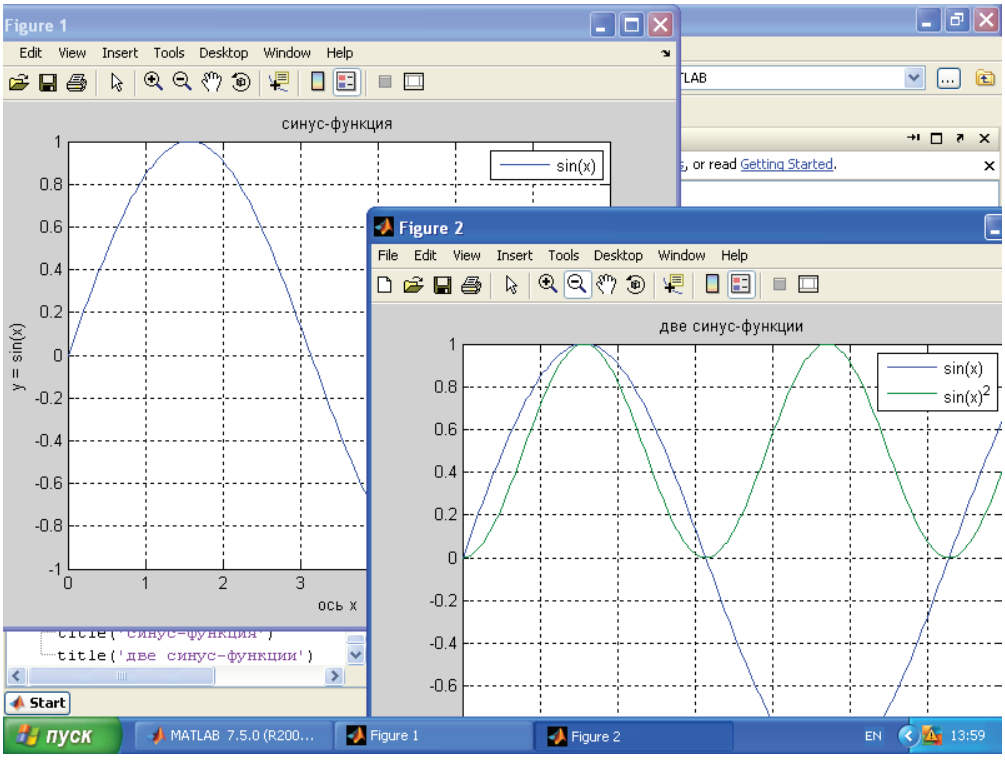


Рис. 3.1. Примеры окон с графиками функций

Окна графиков имеют меню, с помощью которого эти графики можно распечатывать или вставлять в тексты, создаваемые с помощью программ-редакторов.

Примеры построения графиков функций

`>>x=[0:pi/90:2*pi+0.733];`	Определяется переменная x с начальным значением 0, конечным значением 2*pi +0.733 и шагом pi/90 между значениями
`>>y=sin(x);`	Вычисление значений функции для заданных ранее значений аргумента, причём значения аргумента задаются в радианах
`>>plot(x, y);`	Открывается окно, в котором появляется график функции y = sin (x) для заданных ранее значений x. Если же существовало ранее открытое окно, то существующий график будет заменён
`>>grid;`	В окне появляется сетка (или ранее существовавшая сетка убирается)
`>>legend ('sin(x)');`	Вставляется легенда (надпись на графике в правом верхнем углу)
`>>title('синус-функция');`	Вставляется заголовок графика
`>>xlabel('ось x');`	Вставляется надпись для оси x
`>ylabel('y = sin (x)');`	Вставляется надпись для оси y
`>>z=y.*y;`	Вычисление функции sin (x)*sin (x)
`>>figure;`	Открывается новое окно для графика
`>>plot(x, y, x, z); grid;`	В новом окне строятся графики функций y=sin (x), y=sin (x)*sin (x) и наносится сетка
`>>legend('sin','sin^2');`	Вставляется легенда. Если кривых несколько, то надписи отделяются друг от друга запятой
`>>hold;`	Командой **hold** или **hold on** сохраняется текущий график со всеми осями, сетками, заголовками, легендами и другими атрибутами. Последующие кривые после команды plot будут добавляться к уже существующим. Повторным применением команды **hold**, например, команды **hold on**, осуществляется переход в стандартный режим, так что последующая команда **plot** полностью заменяет ранее существовавший график
`>>plot(x, cos(x));`	После команды **hold** в ранее существовавшем окне появится график функции y = cos (x), при этом сохраняются все прежние установки
`>>legend('sin(x)', 'sin(x)^2','cos(x)');`	Расширение легенды добавлением надписи о функции косинус

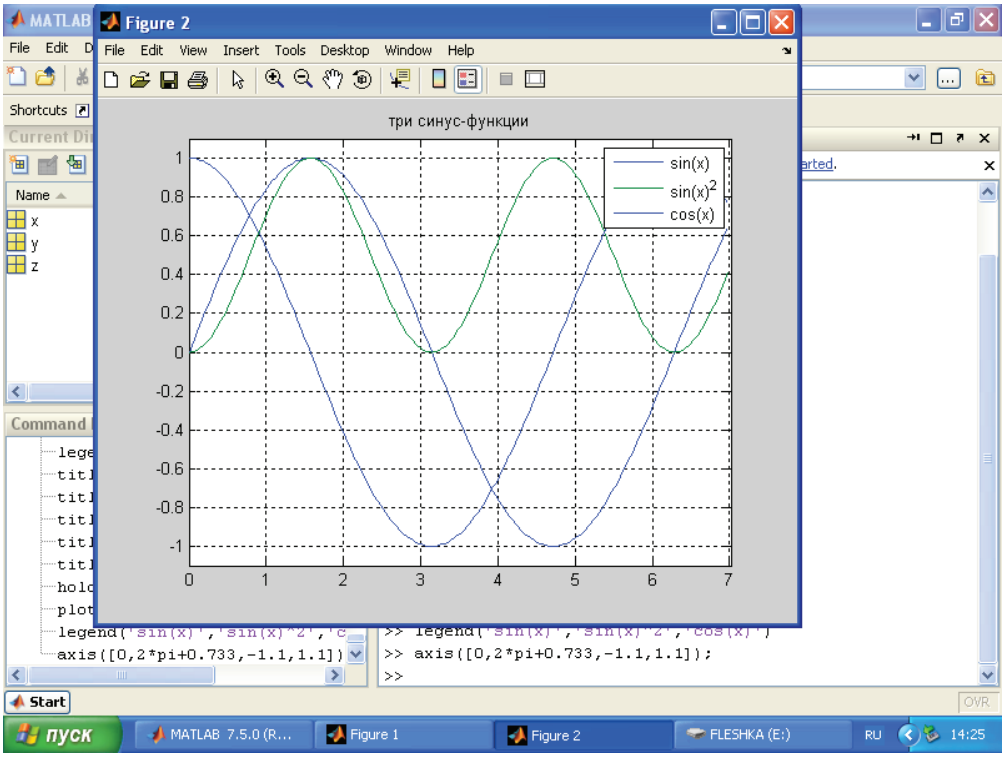


Рис. 3.2. Графики функций y=sin (x), y=sin (x)^2 и y=cos (x)

Типы возможного масштабирования осей

>>plot(x,y) — обычный (линейный) масштаб осей.
>>loglog(x,y) — обе оси задаются в логарифмическом масштабе.
>>semilogx — ось x задаётся в логарифмическом масштабе.
>>semilogy — ось y задаётся в логарифмическом масштабе.

Некоторые дополнительные возможности по оформлению графиков

Имеется достаточно много команд, связанных с оформлением графиков. Ниже будут представлены наиболее полезные из них.

>>gtext('Text');	Текст, который позиционируется на графике с помощью мыши

<table>
<tr><td>>>axis([xmin, xmax, ymin, ymax]);</td><td colspan="3">В соответствии с желанием пользователя можно изменять масштабы осей — для этого служит команда axis, причём факторы масштабирования задаются в виде вектора</td></tr>
<tr><td rowspan="3">>>plot(x, y,'y –');</td><td colspan="3">Наконец имеются опции, с помощью которых определяются цвет и символы, изображающие график. Опции определяются в команде plot после задания векторов для осей x и y и заключаются в апострофы. Другие линии или типы точек:</td></tr>
<tr><td>цвет</td><td>тип точки</td><td>тип линии</td></tr>
<tr><td>b — синий
g — зелёный
r — красный
c — бирюзовый
m — малиновый
y — жёлтый
k — чёрный</td><td>. — точка
o — круг
x — крест
+ — плюс
* — звёздочка
s — квадрат
d — ромб
v — треугольник (вниз)
^ — треугольник (вверх)
< — треугольник (влево)
> — треугольник (вправо)
p — пентаграмма (звезда)
h — гексаграмма (звезда)</td><td>- — минус
: — двоеточие
-. — минус и точка
-- — пунктир</td></tr>
<tr><td></td><td colspan="3">Здесь указаны символы, которые изображают точки той или иной кривой</td></tr>
</table>

Другие свойства графика можно изменять при помощи меню графического окна. К примеру, можно поменять масштабы осей через окна **Edit** → **Axes Properties**. MATLAB предлагает и другие возможности графического представления данных, которые здесь не рассматриваются. При необходимости с ними можно познакомиться с помощью команды **help plot**, вызывающей пояснения в командное окно.

Глава 4.
Программирование в MATLAB

Интересно программирование в MATLAB с использованием М-файлов. Операторы языка программирования MATLAB можно выполнять и в составе М-файла, и непосредственно задавать в командном окне.

4.1. Использование редактора М-файлов для создания программ

MATLAB предоставляет в распоряжение пользователя похожий на язык С полнофункциональный язык программирования (М-язык), который позволяет записать последовательность операторов языка и набора данных, а затем запустить их на выполнение с помощью одной команды [1, 2, 4, 5]. Программа, так называемый М-файл, создаётся с помощью редактора М-файлов или с помощью текстового редактора. Имя М-файла должно иметь расширение m (Name.m). М-файл можно создать и через последовательность **`File` → `New` → `M-File`**.

Имя М-файла является новой командой, с помощью которой программу запускают из командного окна MATLAB или SIMULINK. М-файлом может быть и программа, которая фактически представляет собой набор команд MATLAB (скрипт), который удобно использовать при необходимости многократного выполнения одной и той же последовательности команд. В качестве операторов возможно использование функций, преобразующих входные данные в некоторые выходные.

Как уже отмечалось, MATLAB дает возможность писать самостоятельные программные модули на разных языках, однако основными являются C++ и М-язык. Все, что до сих пор делалось в командном

окне, подчиняется синтаксису М-языка системы MATLAB. Язык использует следующие средства:

- данные различного типа;
- константы и переменные;
- операторы, включая операторы математических выражений;
- встроенные команды и функции;
- функции пользователя;
- управляющие структуры;
- системные операторы и функции;
- средства расширения языка.

М-язык интерпретируется, поэтому инструкции программы транслируются и исполняются в порядке записи.

4.2. Основные типы данных

Типы данных в MATLAB структурированы следующим образом:

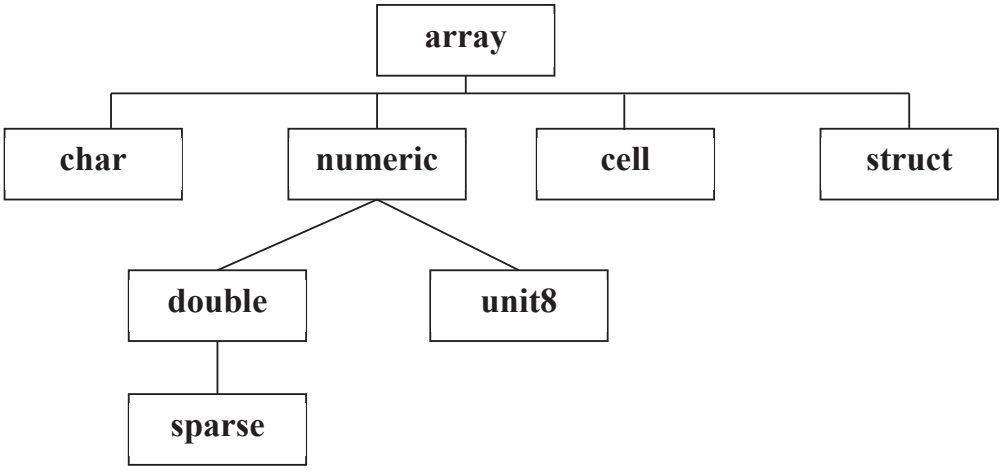


Рис. 4.1. Типы данных в MATLAB

Типы данных **`array`** и **`numeric`** являются виртуальными, поскольку к ним нельзя отнести какие-либо переменные. Они служат для определения и комплектования некоторых типов данных.

Определено шесть типов данных:

- **double** — числовые массивы с числами удвоенной точности;
- **char** — строчные массивы с элементами-символами;
- **sparse** — разреженные матрицы с элементами-числами удвоенной точности;
- **cell** — массивы ячеек, которые в свою очередь могут быть массивами;
- **struct** — массивы записей с полями, которые также могут содержать массивы;
- **unit8** — массивы 8-разрядных целых чисел без знаков (математические операции с ними не предусмотрены).

Существует еще тип данных, определенный пользователем, — **UserObject**.

Чаще всего рассматриваются данные типов **double** и **char**. Описание остальных типов можно посмотреть в разделе **help**. Стоит отметить лишь то, что ячейки типа **cell** представляют собой массивы с элементами разных типов. Для отличия от обычных массивов ячейки заключаются в фигурные скобки.

В иерархии типов данных на самом верхнем уровне находятся данные типа **array**, а это значит, что все виды данных, как уже отмечалось, являются массивами. Типы данных явно не объявляются.

М-язык имеет необходимые средства для различных видов программирования:

- процедурного;
- операторного;
- функционального;
- логического;
- структурного (модульного);
- объектно-ориентированного;
- визуально-ориентированного.

4.3. Модули программ в М-языке

Программные модули М-языка имеют расширение *.m, называются М-файлами и бывают двух типов: Script-файлы (сценарии, управляющие программы) и m-функции (процедуры). И те, и другие имеют расширение *.m, причем при обращении к ним расширение не указывается.

Главным внешним отличием текстов этих двух файлов является то, что у m-функции есть первая строка вида:

```
function[var1, var2, …] = имя функции (par1, par2,..),
```

где **var** — выходные выражения (если оно одно, то скобки могут опускаться), а **par** — входные параметры (аргументы функции).

Такая строка не имеет сценария. Он является просто записью последовательности команд без входных и выходных переменных. Принципиальное отличие заключается в разном восприятии системой имен переменных в этих файлах.

В сценариях все переменные помещаются в рабочее пространство **Workspace**, как и переменные командного окна, поэтому связь с рабочим пространством делает эти файлы удобными для управления вычислительным процессом.

m-функции располагают собственным пространством переменных, изолированным от рабочего пространства системы MATLAB, поэтому совпадение имен из рабочего пространства и имен внутренних переменных m-функций не приводит к противоречиям. Если переменные, которые используются в теле m-функции, не совпадают с именами формальных параметров этой функции, то они называются локальными. Их область действия полностью ограничена рамками тела данной m-функции. Они не видны из рабочего пространства системы MATLAB и из других m-функций.

Основным каналом передачи информации из командного окна в m-функцию и из одной функции в другую является механизм параметров функции. Другим механизмом передачи информации в функцию являются глобальные переменные. Чтобы рабочая область системы MATLAB и (или) несколько m-функций могли совместно использовать некоторую переменную с заданным именем, всюду, где она используется, её объявляют как глобальную с помощью ключевого слова **global**. Локальные переменные не сохраняют своих значений между вызовами функции. При необходимости передачу параметров между вызовами можно организовать с помощью статических переменных, которые объявляются ключевым словом **persistent**. Внутри m-функции могут размещаться другие процедуры. Они подчиняются тем же правилам и располагаются в конце тела основной функции.

MATLAB допускает построение рекурсивных алгоритмов.

4.4. Запись текстов для M-файлов

В одной строке можно размещать несколько операторов, разделяя их символом «;».

Если оператор не размещается в одной строке, то строку необходимо заканчивать тремя точками «...».

Если оператор не имеет в конце символа «;», то результат его работы будет выведен на экран.

Текст, начинающийся символом «%», считается комментарием и не обрабатывается.

Комментарий перед первым исполняемым оператором рассматривается как описание программы и может быть выведен по команде **help <имя файла>**.

В M-языке переменные не описываются и не объявляются. Любое новое имя воспринимается как имя матрицы. Размер этой матрицы устанавливается при предварительном вводе значений её элементов либо определяется действиями по установлению значений её элементов, описанными в предыдущем операторе или процедуре.

Для выхода из программы в произвольной точке служит оператор **return**.

Пример использования MATLAB-команд

M-файл summa.m содержит приведённую ниже последовательность команд.

% summa.m — пример простой программы для вычисления суммы чисел 1,..,n

clc;% очистка экрана

echo off; % все выполняемые команды m-файла будут выданы на экран

%diary('Primer.doc');% Протоколировать вводимые и выводимые данные в документе

clear n sum i tast; % Обнуление переменных

n=input('Задайте,пожалуйста, целое число:'); % Задание числа n

% Проверка правильности задания n:

```
  if isempty(n) % Выдаёт «истина», т.е. 1, если n — пустая матрица.
  disp('Ошибка ввода. Пожалуйста, запустите программу
сначала.'),tast=input(' ');
  return; % Назад в командное окно MATLABa откуда стартовали
  elseif  n==0 % n=0
  disp('Пожалуйста стартуйте вновь задав другое число'),
tast=input(' ');
  return; % Назад в командное окно MATLABa, где был запущен
M- файл
  else
      sum=0;
      for i=1:n
          sum=sum+i;
      end
  end
  disp('Сумма чисел от 1 до n составляет:'),sum
  echo off; % Выполненные команды не будут больше выдаваться
```

Вызов и выполнение команд приведённого в качестве примера M-файла из окна MATLAB осуществляется командой summa.

4.5. Основные операторы MATLAB

С помощью операторов MATLAB записывают алгоритмы. Примером операторов могут служить операторы ввода-вывода, циклы или условные операторы.

Операторы ввода-вывода

`>>D=input('<Текст>')`	Оператор используется для ввода-вывода данных. **<Текст>** — сопровождающий ввод текста комментарий

`>>D=input('Длина=')`
`>>Длина = 8.9`
`>>D`
`D=8.9`
	Пример использования оператора

`>>D=input('<Текст>','s')`	Второй вид оператора ввода. Здесь `'s'` — параметр функции input, указывающий, что будет введено строковое выражение

`>>E=input ('Введите выражение', 's')`
`>>Введите выражение 2*sin(pi/6)`
`>>E =`
`2*sin(pi/6)`
`>>eval(E)`
`ans=`
`1.0000`
	Пример использования второго вида оператора. Здесь `'s'` — второй параметр, который указывает, что будет введено строковое выражение. Функция **eval** преобразует символьное выражение в числовое и делает вычисление

`>>disp(A)`	Оператор вывода результатов; записывается с единственным аргументом, который может быть числовым или символьным массивом, поэтому при необходимости вывести несколько переменных их объединяют в массив

Циклы for

`>>for i=<Выражение>`
`<Инструкции>`
`end;`
	Если предполагается многократно выполнять одну и ту же последовательность операторов, то в таких случаях можно использовать оператор цикла типа **for**. Цикл **for** всегда должен завершаться словом **end**. <**Инструкции**> могут включать в себя несколько операторов, которые при этом не заключаются в скобки. Принципиальное отличие рассматриваемого языка от других языков в том, что в цикле **for** <**Выражение**> является матрицей. При каждой итерации значения столбцов матрицы присваиваются переменной i

`>> A=[1 2; 4 5]`
`A = 1 2`
`4 5`
`>>for i = A`
`x = i`
`end;`
`x = 1`
`4`
`x = 2`
`5`
	Пример цикла **for**.
	Возможность1: <**Выражение**> является матрицей. Переменной i при каждой итерации присваиваются значения столбцов матрицы. Такая конструкция очень удобна при работе с матрицами

`>> a = 1:3`
`a = 1 2 3`
`>>for i = a`
`x = i`
`end;`
`x = 1`
`x = 2`
`x = 3`
	Возможность 2: **<Выражение>** является вектором. Здесь a является вектором-строкой, т. е. матрица имеет в столбце только один элемент. Снова в цикле **for** при каждой итерации переменной i присваивается значение вектора-столбца, в данном случае — скаляра. Этот цикл соответствует привычным циклам, известным из других языков программирования

Циклы while

`>> while B`
`<Операторы>`
`end;`
	В этом цикле B — логическое условие: пока B — истина (**true**), последовательность **<Операторы>** выполняется. В отличие от других языков программирования, здесь B — матрица. И до тех пор, пока все элементы матрицы не равны нулю, цикл будет выполняться

`>> A = [1 2; 3 4]`
`A = 1 2`
`3 4`
`>>while A`
`A(1,2)= A(1,2)– 1`
`end;`
`A = 1 1`
`3 4`
`A = 1 0`
`3 4`
	Здесь B = A — матрица, и только после двойного прохода, когда один элемент матрицы обнуляется, цикл прерывается. Более привычным является применение цикла, когда B — скаляр, т. е. в таком случае матрица представлена одним элементом. В большинстве языков программирования B — логическое или арифметическое выражение

`>>n = 3; x = 1;`
`>>while n > 0`
`n = n – x`
`end;`
`n = 2`
`n = 1`
`n = 0`
	В этом примере B — скаляр (простая переменная)

Условный логический оператор

>>if B1
<Операторы1>
elseif B2
<Операторы2>
else
<Операторы3>
end;
	Каждый такой оператор должен заканчиваться словом **end**. Выполняется этот оператор так же, как и в других языках программирования, т. е. B1 и B2 являются логическими условиями, но, как и в предыдущем случае, они — матрицы. Если, например, все элементы матрицы B1 не равны нулю, то выполняются **<Операторы1>**, а **<Операторы2>** и **<Операторы3>** пропускаются

>>A = [1 1; 3 4]
A = 1 1
3 4
>>if A
A(1,2)=A(1,2) – 1
else
A(1,2) = A(1,2) + 2
end;
A = 1 0
3 4
>>if A
A(1,2)=A(1,2) – 1
else
A(1,2) = A(1,2) + 2
end;
A = 1 2
3 4
	В начале выполняется первый оператор, поскольку все элементы матрицы ненулевые. При дальнейшей работе будет выполнен второй оператор. Обычно же условие B1 бывает скаляром, а не матрицей

4.6. Функции в MATLAB

В MATLAB можно запрограммировать собственные функции [2–4]. Эти функции хранятся как отдельные M-файлы, при этом первым словом M-файла должно быть слово function. В функцию могут быть переданы аргументы. Определённые внутри функции переменные являются локальными и не хранятся на поверхности рабочего стола. С помощью функций можно создавать новые библиотеки команд.

Синтаксис функции

```
>>function [Выходные аргументы]= Имя функции (Входные аргументы) <операторы>
```

Пример функции

Функциям можно передавать различные входные величины, при этом оказывается возможным гибкое применение функций, когда равные операции применяются для различных переменных. В SIMULINK функции могут выдавать последовательности выходных величин.

```
function y = stat(u)
```

% функция **y=stat(u)** — простая функция для вычисления средних величин и среднеквадратических отклонений элементов вектора u.

% функция вызывается оператором **stat(Varbl)**, причём **Varbl** является любым ранее определённым вектором.

```
n = length (u);
mean = sum (u)/n;
stdev = sqrt (sum ((u-mean).^2/n));
y= [mean, stdev];
disp ('первая величина – среднее значение элементов вектора x')
disp ('вторая величина – среднеквадратическое отклонение')
```

Вызов осуществляется в командном окне MATLAB после того, как вектор x будет определён в качестве входной величины:

```
>>x = [3 5 3 5 7]; y= stat (x).
```

Отсюда ясно, что в MATLAB, как и в других языках программирования, наряду со стандартными операторами возможно определение и последующее использование функций для выполнения сложных математических вычислений.

Глава 5.
Введение в Control Toolbox (команды и инструменты, применяемые в области теории управления)

5.1. Передаточная функция контура регулирования

Пример передаточной функции $G_s(s) = \frac{K_s}{1 + sT_s}$, в общем случае:

$$G_s = \frac{a_n s^n + \ldots + a_2 s^2 + a_1 s^1 + a_0}{b_m s^m + \ldots + b_2 s^2 + b_1 s^1 + b_0}.$$

Задание и представление полиномов в MATLAB

`>>num = [2 1 0]`
`num = 2 1 0`
`>>den = [3 2 1]`
`den = 3 2 1`
	Полином $a_n s^n + \ldots + a_2 s^2 + a_1 s^1 + a_0$ — это вектор, элементы которого в нисходящем порядке являются коэффициентами полинома

`>>Gs=tf(num, den)`
`Transfer function:`
`  2s^2+s`
`---------`
`3s^2+2s+1`
	Для передаточной функции, состоящей из числителя и знаменателя, числитель (от англ. numerator) и знаменатель (от англ. denumerator) задаются в виде отдельных векторов-строк. Командой **`tf`** (числитель, знаменатель, от англ. transfer function) передаточная функция задаётся в привычной форме

>>Ks=5; Ts=3; **>>Gs2=tf(Ks,[Ts,1])** **Transfer function:** **5** **———** **3s+1**	Векторы-строки могут также использоваться внутри круглых скобок команды **tf** для задания передаточных функций, как это показано здесь при задании передаточной функции апериодического звена
>>num = conv([1 5], [1 6]) **num = 1 11 30** **>>den = conv([1 6 10], [1 3])** **den = 1 9 28 30** **>>Gs = tf(num, den)** **Transfer function:** **s^2+11s+30** **————————————** **s^3+9s^2+28s+30**	Задание передаточной функции как комбинации полиномов, если, например, известны нули и полюса: $G_s(s)=\frac{(s+5)(s+6)}{(s+3)(s^2+6s+10)}$. При этом команда **conv (Polynom1, Polynom2)** перемножает два полинома и представляет результат в виде вектора-строки с элементами, представляющими соответствующие коэффициенты полинома-произведения в таком порядке, как они представлены в результирующем полиноме: $G_s(s)=\frac{s^2+11s+30}{s^3+9s^2+28s+30}$

Преобразование и последующее представление передаточной функции через нули и полюса

>>G=zpk (Gs) **Zero/pole/gain:** $\frac{(s+5)(s+6)}{(s+3)(s^2+6s+10)}$	Команда преобразует передаточную функцию так, что её полиномы оказываются разложенными на элементарные сомножители. Достоинство преобразования в том, что сразу становится видна компенсация полюсов знаменателя нулями числителя в каком-либо регуляторе, а полином второго порядка указывает на наличие комплексно-сопряжённых корней

5.2. Графические возможности представления передаточных функций

Обзор основных возможностей графического представления

<table>
<tr><td>>>nyquist(Gs);</td><td>Годограф кривой Найквиста. Нежелательное представление кривой в области отрицательных частот может быть предотвращено щелчком правой клавиши мыши на свободной поверхности рядом с кривой Найквиста в окне Show –> Negative Frequencies и снятием галочки</td></tr>
<tr><td>>>step(Gs);</td><td>Реакция на единичную функцию (переходная функция)</td></tr>
<tr><td>>>impulse(Gs);</td><td>Реакция на дельта-функцию (функция веса)</td></tr>
<tr><td>>>bode(Gs);</td><td>Логарифмические амплитудно-частотные и фазочастотные характеристики (диаграмма Боде)</td></tr>
<tr><td>>>w=tf([1,[4 0.5 1]);
>>g=tf(1,[1 1 1]);
>>subplot(2,2,1);
>>nyquist(w, g); grid;</td><td rowspan="4">Команда subplot открывает графическое окно, в котором резервируется место, например, для одного из четырёх графиков (см. рис. 5.1). Размещение в два ряда по два окна в каждом ряду: первая цифра — количество рядов, вторая — количество окон в ряду. Последняя цифра в скобках даёт место, на котором будет размещён следующий график. Графики пронумерованы в направлении слева направо и сверху вниз. Нужно иметь в виду, что по мере увеличения количества окон размеры графиков уменьшаются</td></tr>
<tr><td>>>w=tf([1,[2 1 1]);
>>subplot(2,2,2);
>>step(w, g); grid;</td></tr>
<tr><td>>>subplot(2,2,3);
>>impulse(w, g); grid;</td></tr>
<tr><td>>>w=tf(150,[1 2]);
>>g=tf(1,[4 0.2 1]);
>>subplot(2,2,4);
>>bode(w, g); grid;</td></tr>
<tr><td>>>pzmap(Gs);</td><td>Команда вычисляет нули (0) и полюса (x) передаточной функции (G_s) и представляет их на графике (см. рис. 5.2)</td></tr>
</table>

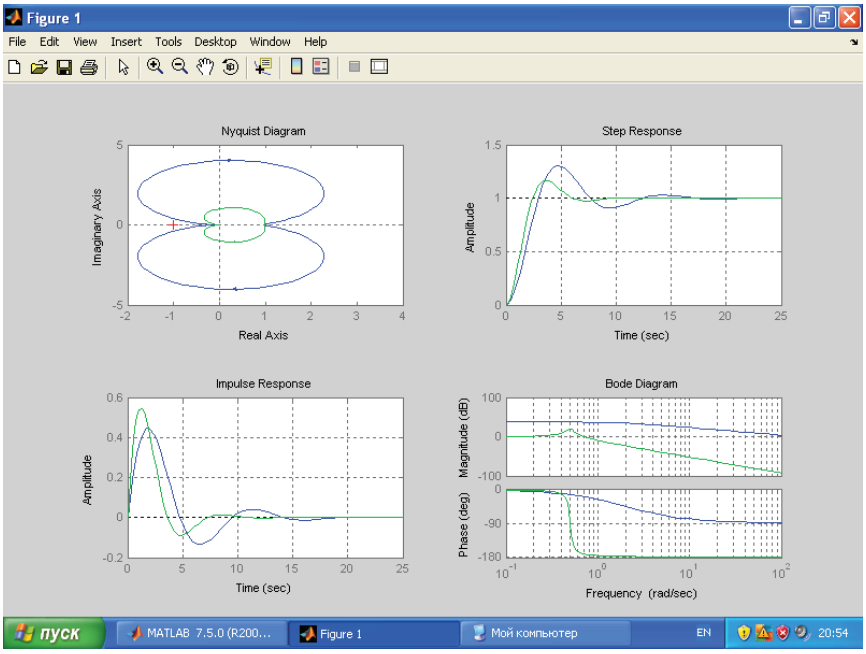


Рис. 5.1. Примеры графиков (команды **nyquist**, **step**, **impulse**, **bode**) для различных передаточных функций с использованием команды **subplot**

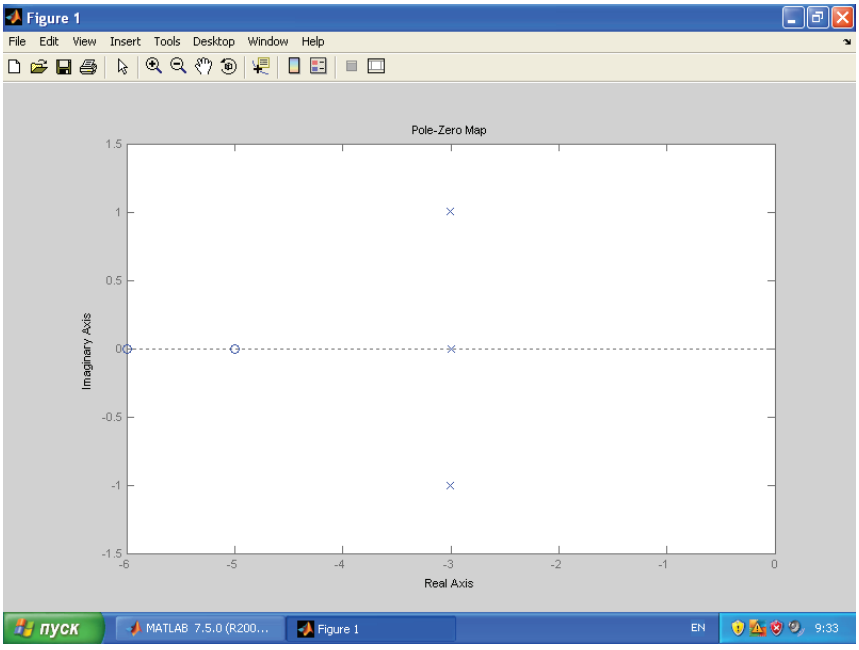


Рис. 5.2. Нули и полюса передаточной функции (команда **pzmap**)

5.3. Характеристики передаточной функции

Вычисление нулей и полюсов передаточной функции

`>>Ps = pole(Gs);`
`Ps = -3.0000 + 1.0000i`
`-3.0000 - 1.0000i`
`-3.0000`
	Вычисление полюсов передаточной функции

`>>Ns = tzero(Gs);`
`Ns = -6.0000`
`-5.0000`
	Вычисление нулей передаточной функции. Для этой цели можно применять также команду `tzero(Gs)`

5.4. Соединения блоков

Передаточные функции при наличии обратной связи

`>>G= feedback(Gs,1)`
`Transfer function:`
`s^2+11s+30`
`————————`
`s^3+10s^2+39s+60`
	Передаточная функция G_s с отрицательной обратной связью:
	$G = \frac{G_s}{1+G_s}$

`>>Gs1=tf([1 1],[1 0]);`
`>>Gs2=tf([1 0],[1 0 1]);`
`>>Gs=feedback(Gs1,Gs2);`
`Transfer function:`
`s^3+s^2+s+1`
`——————`
`s^3+s^2+2s`
	Передаточная функция G_s при наличии в прямой цепи функции G_{s1}, а в цепи обратной связи — функции G_{s2}

`>>Gs=feedback(Gs1,Gs2,+1)`
`Transfer function:`
`s^3+s^2+s+1`
`————————`
`s^3-s^2`
	Если речь идёт о положительной обратной связи, то необходимо поставить в скобках +1

Последовательное соединение

`>>Gs_ser1=series(Gs1,Gs2)`
`Transfer function:`
`s^2+s`
`------`
`s^3+s`
	Командой **series(**G_{s1}**,**G_{s2}**)** можно представить передаточную функцию как произведение двух других

`>>Gs_ser2=Gs1*Gs2*Gs3`	Знак умножения делает то же самое, но при этом возможно представить последовательное включение трёх и более передаточных функций

Параллельное включение

`>>Gs=parallel(Gs1,Gs2)`
`Transfer function:`
`s^3+2s^2+s+1`
`--------------`
`s^3+s`
	Командой **parallel(**G_{s1}**,**G_{s2}**)** можно представить передаточную функцию, состоящую из двух параллельных передаточных функций

`>>Gpi=parallel(tf(1,[1 0]),5)`
`Transfer function:`
`5s+1`
`----`
`s`
	Передаточная функция пропорционально-интегрального регулятора, состоящая из пропорциональной и интегральной части:
	$G_{PI}=\frac{1}{s}+5=\frac{5s+1}{s}$

`>>Gs=Gs1+Gs2+Gs3+Gs4;`	Знак сложения даёт возможность объединить несколько передаточных функций

`>>Gpid=5+tf(1,[1 0])+`
`tf([1 0],1)`
`Transfer function:`
`s^2+5s+1`
`-------`
`s`
	Пример регулятора, состоящего из пропорциональной, интегральной и дифференциальной части:
	$G_{PID}=5+\frac{1}{S}+s=\frac{5s}{s}+\frac{1}{s}+\frac{s^2}{s}=\frac{s^2+5s+1}{s}$

Глава 6.
Введение в SIMULINK

6.1. Начальные сведения о SIMULINK

SIMULINK — специальный набор инструментов MATLAB, с помощью которого можно моделировать логические схемы, системы управления или поведение контуров регулирования [2, 3, 5]. SIMULINK позволяет графически составлять модели из ранее составленных или самостоятельно определённых блоков, которые являются M-файлами. Вызов SIMULINK осуществляется в командном окне MATLAB заданием команды **>>Simulink** или через пиктограмму  . После этого открывается отдельное окно **Simulink Library Browser** (рис. 6.2, слева), в котором представлен обзор всех имеющихся в SIMULINK блоков, а также подготовленных ранее моделей.

Для создания новой модели нужно пройти по цепочке **File** → **New** → **Model** и открыть собственное пустое окно (рис. 6.1).

Окно библиотеки SIMULINK **Simulink Library Browser** (см. рис. 6.2, слева) содержит меню и иконки различных блоков библиотеки. Важнейшими из них для решения задач в области автоматического регулирования являются следующие блоки MATLAB: **Simulink**, **Control System Toolbox** и **Simulink Extras**. В дальнейшем будут рассмотрены некоторые приёмы работы с ними.

Краткое описание блоков и вставляемых параметров, например, коэффициента усиления, возможно с помощью двойного щелчка внутри выбранного блока. Отдельные блоки перетаскиваются на рабочую поверхность с помощью левой клавиши мыши. Блоки объединяются соединительными линиями или с помощью левой клавиши от отмеченного выхода к отмеченному входу следующего блока или с помощью правой клавиши мыши, которая позволяет соединять друг с другом отдельные линии и блоки.

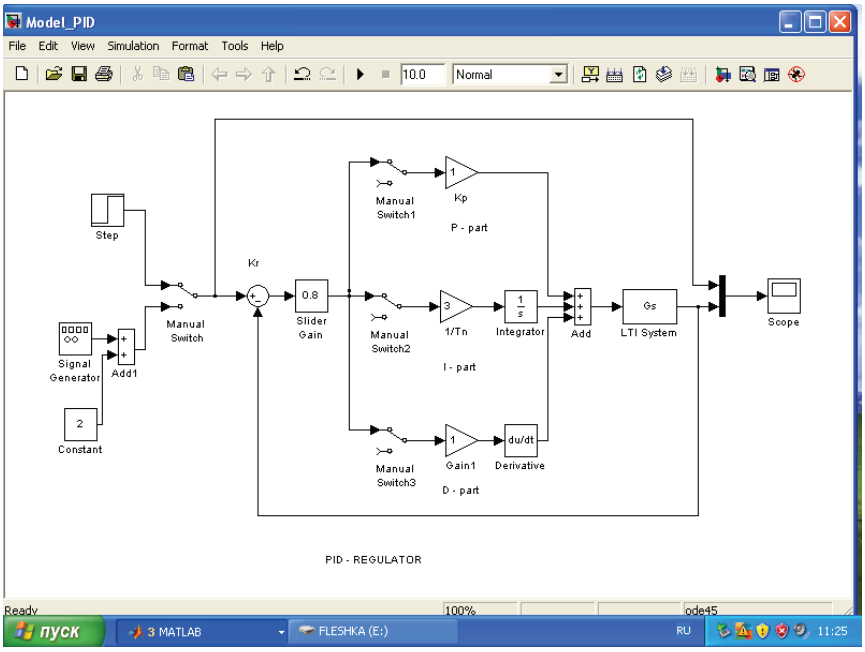


Рис. 6.1. Пример моделирования системы регулирования

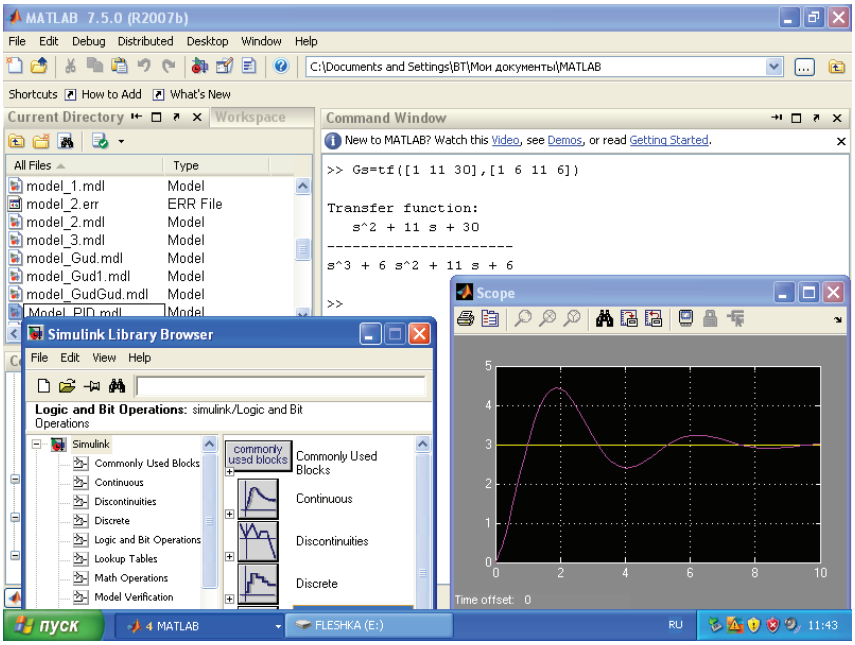


Рис. 6.2. Реакция регулятора на ступенчатое входное воздействие (модель рис. 6.1) и библиотека браузера SIMULINK

6.2. Краткое описание важнейших блоков SIMULINK

Таблица 6.1

Входные величины — источники сигналов (`Simulink` → `Sources`)

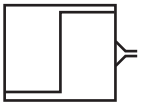
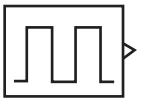
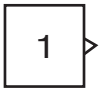
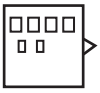
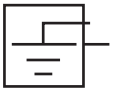
Обозначение блока	Название блока	Назначение блока
	**`Step`**	Регулируемый по высоте скачок (**`Final Value`**) и определяемое начало времени скачка (**`Step Time`**)
	**`Pulse Generator`**	Генератор прямоугольных импульсов (меандр), максимальное значение и частота которых может устанавливаться
	**`Constant`**	Устанавливаемая константа, не изменяющаяся во время моделирования
	**`Signal Generator`**	Генератор для задания различных входных сигналов в виде функций синуса, пилообразной, меандра и других с устанавливаемыми значениями амплитуды и частоты
	**`Clock`**	Генерирует на каждом шаге сигнал, значение которого равно текущему времени моделирования
	**`Ground`**	Позволяет избегать выдачи ошибочных сообщений, если не используемые входы не объединены с другими блоками

Таблица 6.2

Приёмники выходных величин (`Simulink` → `Sinks`)

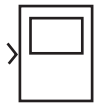
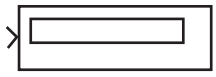
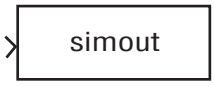
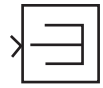
Обозначение блока	Название блока	Назначение блока
 Scope	**`Scope`**	Графическая выдача величин на монитор, похожий на осциллограф. Снимаемые величины в виде переменных могут быть дополнительно сохранены на рабочей поверхности MATLAB. Имя переменной можно выбрать при помощи опции **`Data History`** в окне **`Scope Parameters`**, которое открывается после двойного клика на мониторе **`Scope`** и пиктограмме **`Parameters`**
 Display	**`Display`**	Числовая выдача величин
 To Workspase	**`ToWorkspace`**	Выдаваемые величины могут быть записаны в выбранные переменные, которые можно обрабатывать, используя рабочую поверхность MATLAB, но эти переменные нельзя сохранять. Для считывания переменных можно использовать блок **`From Workspase`** библиотеки **`Sources`**
 To File	**`ToFile`**	Выходная величина может быть сохранена в виде вектора-строки в mat-файле
 Terminator	**`Terminator`**	Похож на блок **`Ground`**, применяется для исключения ошибочных сообщений, если выход блока не соединён с другим блоком

Таблица 6.3

Элементарные блоки аналоговых систем регулирования (`Simulink` → `Continuous`)

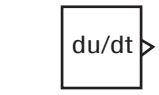
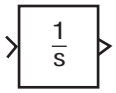
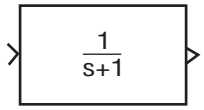
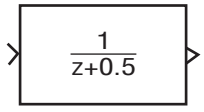
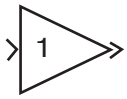
Обозначение блока	Название блока	Назначение блока
 Derivative	`Derivative`	Дифференцирующее звено (дифференциатор), но без параметров, поэтому коэффициент усиления должен быть дополнительно предусмотрен с помощью отдельного блока
 Integrator	`Integrator`	Интегрирующее звено (интегратор), у которого постоянная времени тоже должна быть дополнительно предусмотрена. Можно определить начальные значения выходной величины (`Initial condition`), а также её ограничения
 Transfer Fcn  Transfer Fcn	`Transfer Function`	Передаточная функция, у которой числитель (`Numerator`) и знаменатель (`Denumerator`) могут быть заданы в виде нисходящей последовательности коэффициентов как векторы-строки (сравните с командой MATLAB `tf`). Возможно также использование уже заданных на рабочей поверхности MATLAB переменных для определения числителя и знаменателя. Кроме того, необходимо следить за тем, чтобы порядок числителя был меньше порядка знаменателя. Начальные значения предполагаются нулевыми.

Таблица 6.4

Некоторые математические операции (`Simulink` → `Math Operations`)

Обозначение блока	Название блока	Назначение блока
 Gain	`Gain`	Коэффициент усиления (множитель) или пропорциональный регулятор. Может применяться для поэлементного умножения, если в качестве сомножителя указывается вектор. Возможно применение в матричных операциях умножения

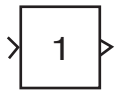
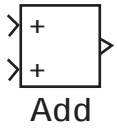
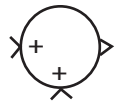
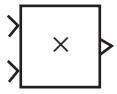
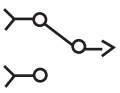
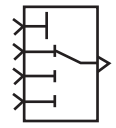
 Slider Gain	**Slider Gain**	Коэффициент усиления или множитель, который можно менять во время моделирования. В отличие от блока Gain, в нём возможно задание только скалярных величин. Часто его применяют для того, чтобы увеличить коэффициент усиления привести контур регулирования к границе устойчивости
 Add  Sum	**Sum**	Сложение или вычитание различных сигналов, необходимых для реализации обратных связей в контурах регулирования или при параллельном включении различных блоков (ПИД-регулятор). Форма (круглая или прямоугольная) выбирается в зависимости от количества и расположения знаков сигналов. Блок можно использовать для сложения векторных сигналов; размерности суммируемых сигналов при этом должны совпадать
 Product	**Product**	Блок можно использовать для умножения (деления) скалярных, векторных и матричных сигналов. В общем случае необходимо следить за соответствием размерностей обрабатываемых величин.

Таблица 6.5

Маршрутизация сигналов (`Simulink → Signal Routing`)

Обозначение блока	Название блока	Назначение блока
 Manual Switch	**Manual Switch**	Переключатель между двумя возможными состояниями с помощью двойного щелчка левой кнопкой мыши. Может применяться, например, для подключения или отключения тех или иных регуляторов. Не подключённые входы должны быть заземлены (блок **Ground**)
 Multiport Switch	**Multiport Switch**	Пропускает на выход в зависимости от значения управляющего сигнала (самый верхний вход слева) тот или иной входной сигнал. Номер пропускающего порта при этом равен значению управляющего сигнала, значение же управляющего сигнала может округляться до целого. Нумерация входов сверху вниз, начиная с 1. Число входов можно задавать

	**Mux**	Объединяет несколько сигналов в вектор, например, чтобы можно было сравнить входную и выходную величину регулятора (см. рис. 6.2, справа)
	**Demux**	Расщепляет сигнал-вектор на отдельные составляющие-скаляры или векторы меньшей размерности, которые поступают как общий сигнал

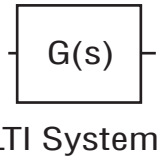
Таблица 6.6

Регулятор с P, I, и D частями (`Simulink Extras` → `Additional Linear`)

Обозначение блока	Название блока	Назначение блока
	**PID-Controller**	Блок для ПИД-регулятора, у которого параметры P-, I- и D-частей могут быть заданы отдельно в форме P + I/s + D*s, т. е. как коэффициенты усиления.

Таблица 6.7

`Control System Toolbox`

Обозначение блока	Название блока	Назначение блока
	**LTI Systems**	Универсально применяемый блок для задания передаточных функций или систем. Вместо передаточной функции в известной форме, когда числитель и знаменатель задаются как векторы-строки коэффициентов, следующим в нисходящем порядке степеням можно задавать имена передаточных функций (G_s), определённых на рабочем столе. Здесь также действует правило: порядок числителя должен быть меньше или равен порядку знаменателя.

Здесь приведены только наиболее часто встречаемые SIMULINK-блоки. Описание других блоков на английском языке можно получить через двойной клик мышью на интересующем блоке.

6.3. Моделирование в SIMULINK

В качестве примера рассматривается регулятор, приведённый на рис. 6.1 с уже определённой в разделе 6.1 передаточной функцией:

$$G_s(s) = \frac{s^2 + 11s + 30}{s^3 + 6s^2 + 11s + 6}.$$

Чтобы избежать сообщения об ошибке, функция $G_s(s)$ должна быть определена на рабочей поверхности MATLAB прежде, чем $G_s(s)$ как параметр будет использована в **`LTI Block`** SIMULINK.

Теперь можно моделировать. Модель, представленная в виде совокупности графических блоков, должна храниться в файле данных с расширением *.mdl. Эти данные содержат лишь информацию о том, какие блоки заданы, какие они имеют параметры и как они соединены. При запуске модели эти данные преобразуются в систему дифференциальных уравнений и записываются в память, поэтому моделирование сводится к численному интегрированию упомянутых дифференциальных уравнений.

Для запуска модели необходимо сначала задать параметры моделирования через меню **`Simulation`** → **`Configuration Parameters`**.

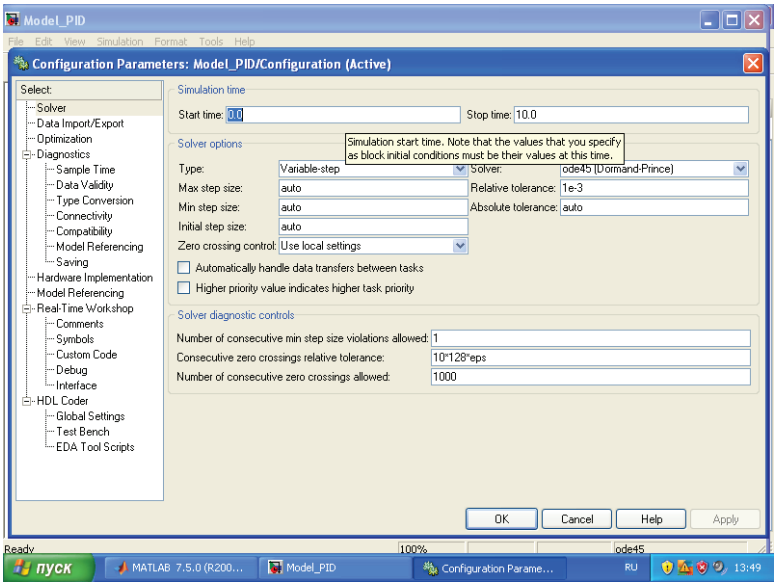


Рис. 6.3. Установка параметров моделирования

Уже отмечалось, что для моделирования в пакете используется численное интегрирование. Подходящий текущей ситуации алгоритм можно задать в приведённом окне, причём обычно используется метод ode-45 (одношаговый метод Рунге — Кутта 4–5 порядка точности), который для большинства моделей, как правило, даёт вполне приемлемые результаты.

Дополнительную информацию о методах численного интегрирования на английском языке можно найти в руководстве по SIMULINK.

Начиная моделирование, необходимо задать время его начала и окончания. Для рассматриваемого примера (рис. 6.1) скачок целесообразно задавать либо в момент времени t = 1 сек., либо t = 0 сек. (в зависимости от сочетания подключённых регуляторов), а время окончания — 10 сек. Возможно, что потребуется задание минимального и максимального шага интегрирования. Обычно модель хорошо работает и при автоматическом выборе шага **auto**.

Модель можно запустить с помощью меню **Simulation** → **Start** или щелчка мыши по пиктограмме старта (треугольник). Двойной щелчок на блоке **Scope** открывает графическое окно, в котором входные и выходные величины выдаются как функции времени. Реакция регулятора на скачок величиной 3 приведена на рис. 6.2.

6.4. Некоторые полезные приёмы при моделировании

Для моделирования реакции системы на скачкообразное воздействие и проверки влияния параметров на эту реакцию (с доведением системы до границы устойчивости) можно применять следующие приёмы:

1) время прекращения моделирования через меню **Simulation** → **Configuration Parameters** установить на величину много большую, чем время переходного процесса (**Stop time** — 150 000, рис. 6.3);

2) вместо входного скачка использовать непрерывный прямоугольный сигнал (меандр), то есть блок **Step** заменить блоком **Signal Generator** со следующими параметрами (через меню **Source Block Parameters: Signal**):

— амплитуда 0,5;

— форма сигнала **square**;

— частота 0,1 Гц (период — 10 сек., рис. 6.4);

3) открыть **Scope** двойным щелчком мыши, а панель **Scope Parameters** — щелчком на второй пиктограмме панели сверху. Интервал времени, обычно установленный на **auto** и выбираемый по обстоятельствам, в данном случае установить на 10 сек.

После проведения этих установок можно запускать модель. Поскольку на панели **Scope** установлена длительность периода меандра, то можно видеть точно один скачок. Для получения скачка с 0 до 1, амплитуда **Signal Generator** установлена на 0,5, а входной сигнал сдвинут вверх на величину 0,5 (рис. 6.4).

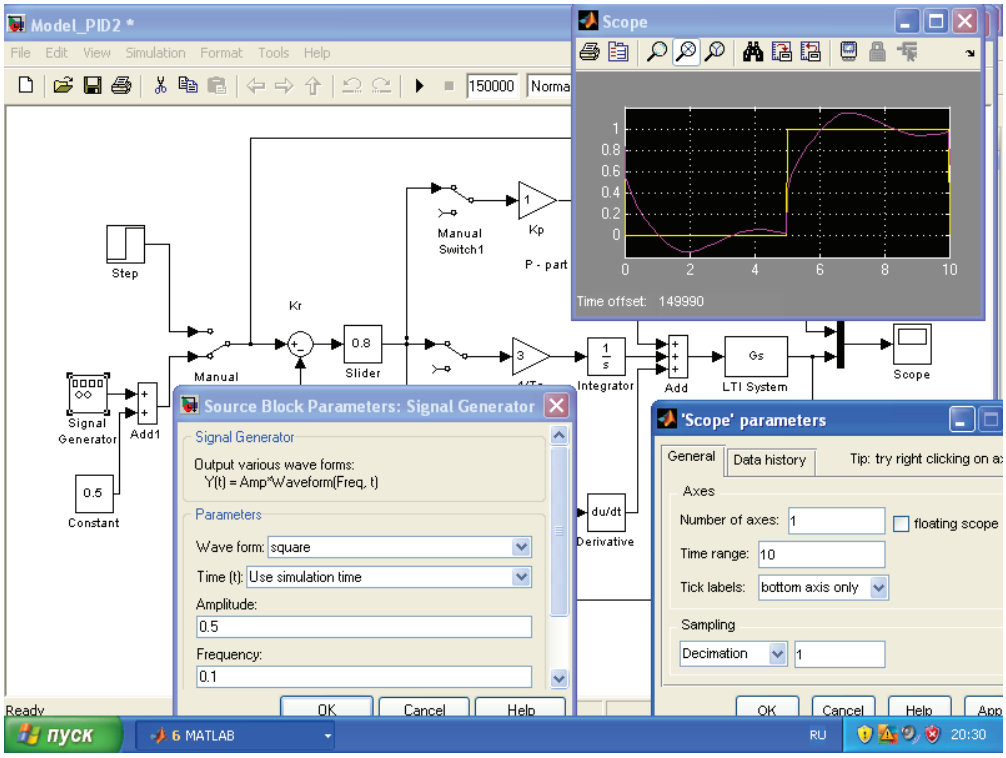


Рис. 6.4. Реакция регулятора на периодическое ступенчатое входное воздействие (рис. 6.1), панели **Scope parameters** и **Signal Generator**

Теперь во время моделирования можно менять текущие коэффициенты усиления блоков регулятора и видеть результат на **Scope** (рис. 6.4).

Если комбинировать варианты подключения различных частей регулятора и менять коэффициенты усиления блоков, то, возможно, придётся задать другую частоту входного сигнала, а в **Scope** выбрать дру-

гой период и даже изменить шаг интегрирования, чтобы можно было проследить поведение регулятора.

D-часть регулятора может приводить к сообщениям об ошибке из-за недостаточно малого шага интегрирования. В этом случае рекомендуется применять блок **PID** из **Simulink-Extras**.

6.5. Использование «осциллографа» (Scope) для представления данных

Результаты моделирования в «осциллографе» всегда представляются на чёрном фоне, который нелегко изменить (это справедливо для MATLAB версий до 6.5). Чтобы уменьшить расход тонера при распечатке большого количества «осциллограмм» и улучшить восприятие рисунка, достаточно изменить фон на более светлый. Сделать это можно двумя способами: в программе обработки изображений или при помощи MATLAB.

1. Изменение графического представления в программах обработки изображений.

Всё содержимое экрана можно загрузить в буфер с помощью клавиши Print Screen и затем в графической программе вставить как новое изображение, а окно «осциллографа» вырезать вручную или использовать функцию некоторых графических программ, которая позволяет копировать окно как изображение.

Как только окно «осциллографа» будет скопировано, можно изменить цвет фона с помощью текущей графической программы, например, поменять местами белый и чёрный цвет фона.

2. Представление содержания окна «осциллографа» (**Scope**) в MATLAB.

Для этого нужно в **Scope parameters** перейти к **Data history**, задать необходимое количество точек графика в окне **Limit data points to last** и выделить галочкой **Save data to workspace**. Необходимо также задать имя для предназначенных к сохранению данных (на рис. 6.5 — **Scope_Data**). Важно выбрать формат данных **Array**, чтобы в дальнейшем в нём можно было обрабатывать данные.

После запуска модели данные сохраняются в переменной **Scope_Data** на рабочем столе (поверхности) MATLAB в виде матрицы. Да-

лее эти данные можно представить в графическом виде с помощью команды **plot** и менять графику в соответствии с пожеланиями и возможностями MATLAB так, как это описано в п. 4.6. При двух выдаваемых величинах (входной и выходной, см. рис. 6.5) соответствующая команда может выглядеть так:

```
>>plot(ScopeData(:, 1), ScopeData(:, 2), ScopeData(:, 1),
ScopeData(:, 3));
grid; title ('Рисунок Scope');
```

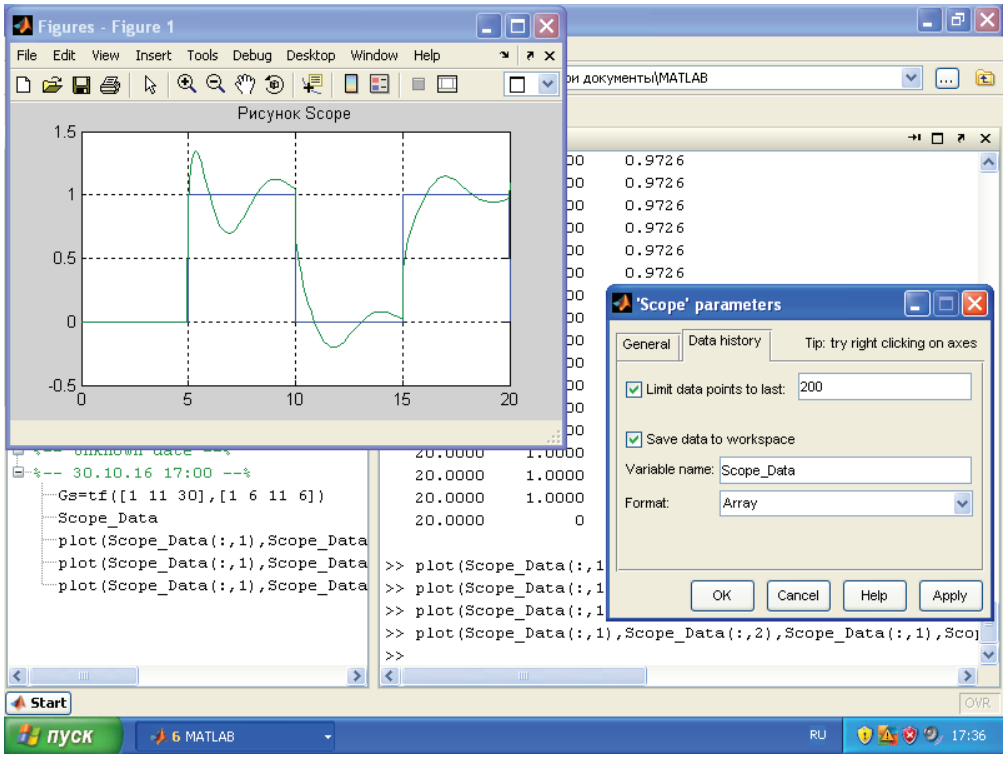


Рис. 6.5. Графическое представление выходных данных модели с помощью команды **plot**

Ясно, что подобным образом можно передавать и сохранять данные результатов моделирования.

Глава 7.
Пример проектирования регулятора с использованием MATLAB

7.1. Определение устойчивости системы регулирования с помощью частотных характеристик (диаграмм Боде)

Как известно, метод исследования устойчивости регулятора по частотным характеристикам является следствием критерия Найквиста и предполагает использование и построение этих характеристик для разомкнутой системы регулирования. Пусть передаточная функция этой разомкнутой системы G_0 представляет из себя последовательное включение двух звеньев: регулятора G_R и объекта регулирования G_S, то есть $G_0 = G_R * G_S$.

Частотные характеристики представляются, как известно, в виде амплитудно-частотной $L(\omega)$ и фазово-частотной $\theta(\omega)$ характеристик, и они имеют общую шкалу частот, а амплитуда $L(\omega)=20*\log_{10}(G_0(j\omega))$ указывается в децибеллах, фаза $\theta(\omega) = \mathrm{Arg}(G_0(j\omega))$ — в градусах. Эти характеристики строятся как функции десятичного логарифма частоты $\log_{10}(\omega)$. В MATLAB диаграмма Боде строится достаточно просто с использованием команды **bode**.

Для суждения об устойчивости цепи регулирования с помощью диаграммы Боде используют два параметра: запас по фазе и запас по модулю. Запас по фазе (англ. phase margin) определяется по фазово-частотной характеристике в точке, где $\log_{10}(G(j\omega)_{dB} = 0$ (здесь $(G(j\omega) = 1)$, а соответствующая частота, обозначаемая как ω_c, называется частотой среза. Запас по модулю (англ. gain margin) определяется на амплитудно-частотной характеристике в точке, где значение фазы равно $-180°$.

Регулятор находится на границе устойчивости, если запасы по модулю ΔL и фазе $\Delta\theta$ нулевые. Принято считать, что контур регулирования оказывается достаточно демпфированным и быстродействующим, если $2 < \Delta L < 6$ и $30^\circ < \Delta\theta < 75^\circ$.

Применение соответствующих команд MATLAB рассматривается на примере разомкнутой системы регулирования, представленной передаточной функцией регулятора

$$G_R = \frac{2,4s + 0,8}{s} = K_C \frac{s + 0,333}{s}$$

и объекта регулирования

$$G_S = \frac{2,5}{0,343s^4 + 1,47s^3 + 2,1s^2 + s}.$$

Передаточная функция всего контура будет $G_0 = G_R * G_S$, а $K_c = 2,4$. Характеристики этого контура представлены на рис. 7.1 и рис. 7.2. Видно, что контур находится на границе устойчивости с практически нулевыми запасами по модулю и фазе.

Регулятор будет устойчивым, если амплитудно-частотную характеристику сдвинуть влево на такую величину, чтобы получившаяся новая частота среза $\omega_{сн}$ обеспечивала запас устойчивости по фазе примерно 60°. Из рассмотрения графиков следует, что в этом случае $\omega_{сн} \approx 0,9$, а амплитудная характеристика опустится примерно на 12 децибелл, и тогда:

$$\log_{10}(K_н) = \log_{10}(K_c) - \Delta K/20,$$

где $\Delta K = 12$, а $K_c = 2,4$ и $K_н \approx 0,6$ — старое и новое (вычисленное по приведённой в предыдущей строке формуле) значения коэффициентов усиления регулятора.

Предполагая, что запас по фазе в 60° и соответствующий запас по модулю при $K_н = 0,6$ обеспечивают удовлетворительное качество процессов, можно записать новую передаточную функцию регулятора:

$$G_{RR} = \frac{0,6s + 0,2}{s}.$$

Соответственно, функция G_0 заменится на другую: $G_{00} = G_{RR} * G_S$. Характеристики этого контура также представлены на рис. 6.6 и 6.7.

Для скорректированного регулятора с использованием команды **`margin`** более точно определены запасы по модулю (12,3 dB) и фазе (65°),

а также построены характеристики, представленные на рис. 7.2 и 7.3. Хорошо заметно, что процессы в контуре с новым регулятором изменились в лучшую сторону.

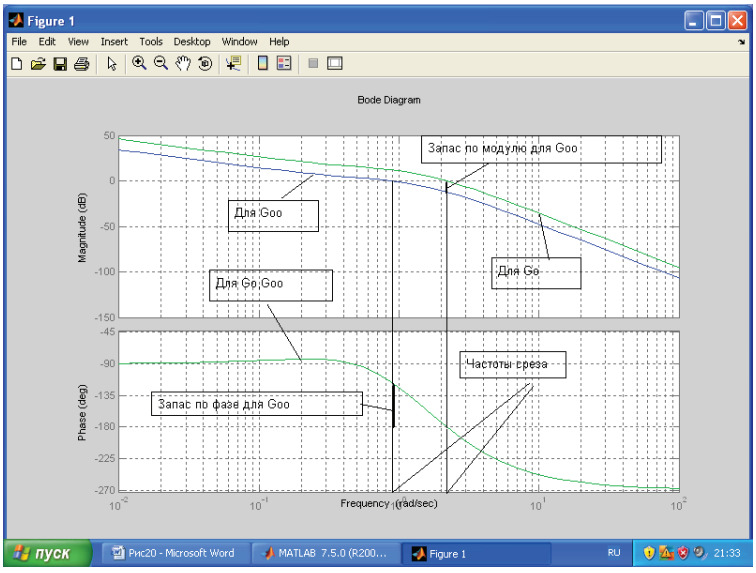


Рис. 7.1. Амплитудно-фазовые и частотно-фазовые характеристики системы (для функций G_0 и G_{00})

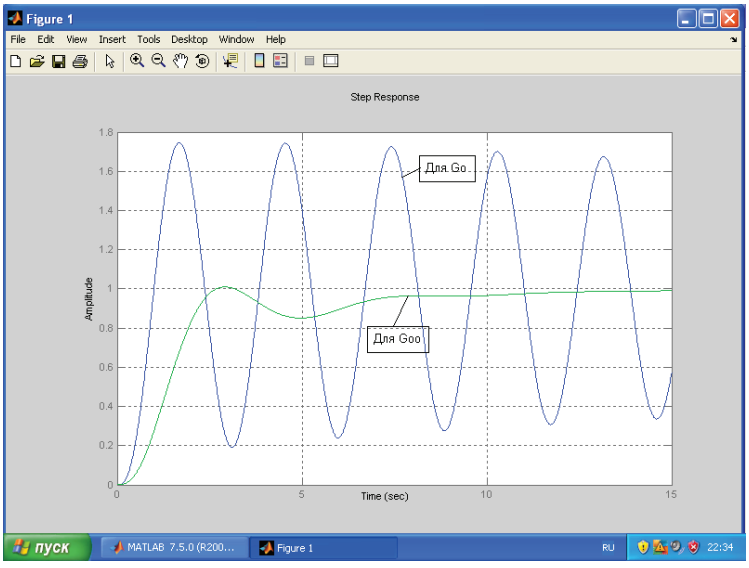


Рис. 7.2. Переходные функции для устойчивой системы (функция G_{00}) и системы на границе устойчивости (функция G_0)

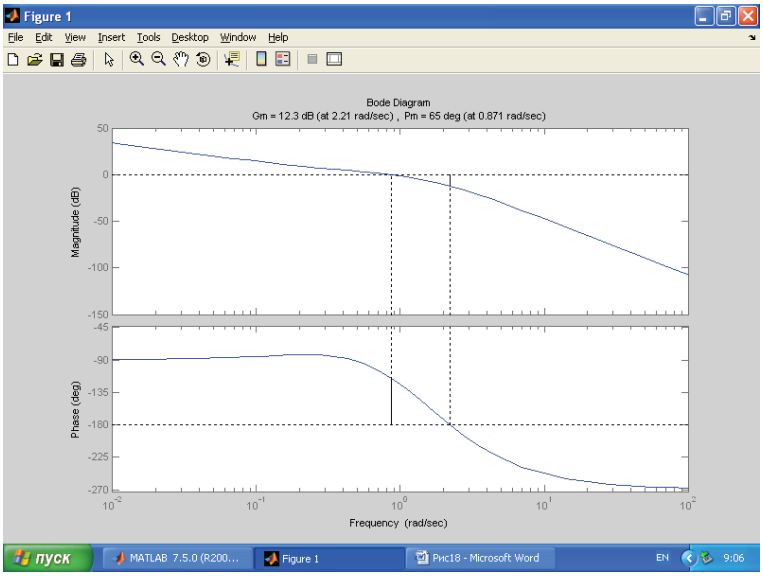


Рис. 7.3. Запасы по модулю (G_m = 12,3 dB) и по фазе (P_m = 65°) скорректированной устойчивой системы (команда **margin**, передаточная функция G_{00})

Примечание. При графическом представлении командой margin MATLAB выдаёт запас по фазе в децибеллах, при численном вычислении запас выдаётся без указания единицы измерения.

Глава 8.
Проектирования регуляторов подчиненного типа

Очень часто современные промышленные системы регулирования строятся на основе принципов подчиненного регулирования [8]. Эти системы отличаются ясностью и простотой синтеза, удовлетворительным качеством работы и возможностью унификации методов синтеза, наладки и блоков аппаратуры систем управления.

8.1. Трехконтурная подчиненная система регулирования (ПСР)

В практике использования ПСР редко бывает больше трёх переменных, поэтому основы теории рассматриваются на примере трёхконтурной системы [8, 10], которая представлена на рис. 8.1.

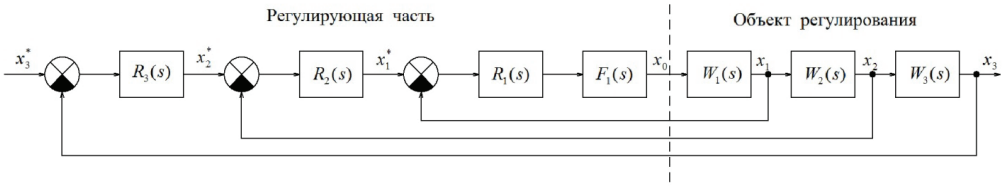


Рис. 8.1. Трехконтурная система подчиненного регулирования

Принцип синтеза ПСР предполагает, что объект регулирования представлен последовательно включенными безинерционными, апериодическими или интегральными звеньями (подобъектами), передаточные функции которых имеют вид:

$$W_i(s),\ i = 1, 2, 3,$$

причём выход каждого из звеньев представлен величиной, которую предполагается регулировать.

Для защиты от помех в цепь каждого из контуров регулирования включают фильтр с передаточной функцией:

$$F_1(s) = \frac{1}{T_\mu s + 1},$$

где T_μ — минимальная (некомпенсируемая) постоянная времени. Каждая регулируемая величина предполагает наличие ее регулятора R_i (s), ($i = 1, 2, 3$).

Структура получившейся системы выглядит как последовательность вложенных друг в друга контуров регулирования [8, 10], причем внешний контур регулирует главную величину (см. рис. 8.1).

8.2. Порядок синтеза регуляторов ПСР

Первым синтезируется самый внутренний регулятор (регулируемая величина x_1), последним — внешний регулятор (регулируемая величина x_3). Это происходит потому, что регулятор каждого контура должен компенсировать инерционность подобъекта регулирования, представленного в данном контуре, и обеспечивать равенство нулю установившейся ошибки (наличие астатизма) [8, 10].

Регулятор первого контура

Типовая структура регулятора первого (самого внутреннего) контура представлена на рис. 8.2. В реальных системах (например, автоматизированный электропривод) этот контур является обычно регулятором либо тока, либо момента.

Исходными данными для конструирования регулятора являются фильтр F_1 (s) и звено объекта регулирования W_1(s) с передаточными функциями:

$$F_1(s) = \frac{1}{T_\mu s + 1},$$

$$W_1(s) = \frac{K_1}{T_1 s + 1},$$

где T_μ — минимальная (некомпенсируемая) постоянная времени; K_1, T_1 — коэффициент усиления и постоянная времени подобъекта регулирования контура.

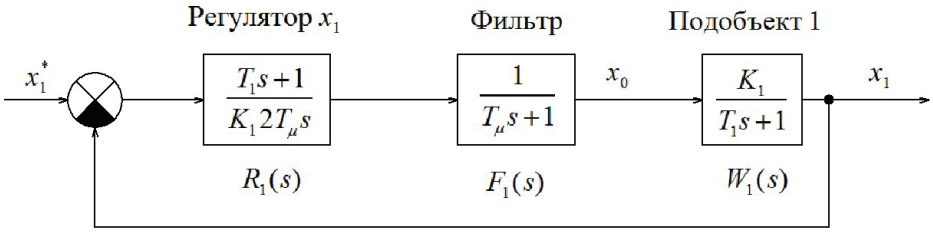


Рис. 8.2. Структурная схема контура регулирования величины x_1

В зависимости от значения параметров знаменателя в выражении для W_1(s), выражение может представлять из себя либо апериодическое, либо интегрирующее, либо усилительное звено.

Регулятор первого контура, как видно из рис. 8.2, имеет вид

$$R_1(s) = \left[W_1(s)\right]^{-1} \frac{1}{2T_\mu s} = \frac{T_1 s + 1}{K_1} \frac{1}{2T_\mu s}.$$

При нулевых начальных условиях переходная функция описывается выражением

$$x_1(t) = 1 - e^{-\tau_1}\left[\cos(\tau_1) + \sin(\tau_1)\right],$$

где $\tau_1 = \frac{t}{2T_\mu}$ — относительное время.

Вид переходной функции показывает, что она не зависит от параметров объекта регулирования, а зависит только от величины минимальной (некомпенсируемой) постоянной времени. Ясно, что это обстоятельство объясняется наличием в регуляторе форсирующего звена, компенсирующего инерционность регулируемого подобъекта. Вариант настройки регулятора, обеспечивающего процесс, приведенный на рис. 8.3, называется настройкой на технический (модульный) оптимум. Иногда такой вариант настройки из-за широкого распространения называют «стандартным», хотя более правильным было бы название «типовой».

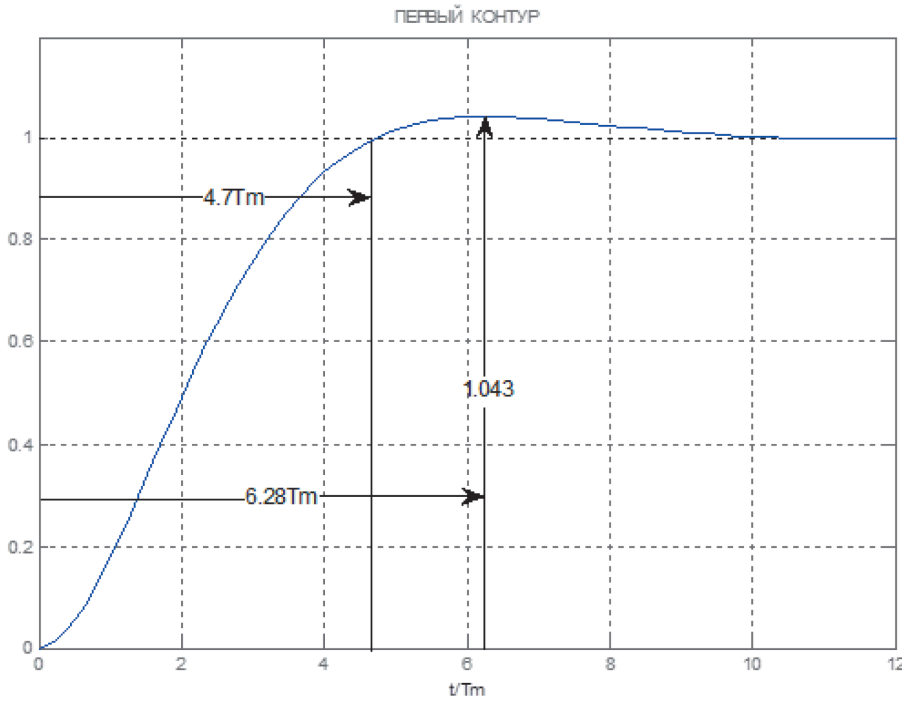


Рис 8.3. Переходная функция регулятора величины x_1

Следующий регулятор можно синтезировать, предварительно определив вид передаточной функции замкнутого регулятора первого контура, для чего сначала необходимо определить вид разомкнутого регулятора:

$$G_1(s) = R_1(s)F_1(s)W_1(s)$$

или

$$G_1(s) = \frac{1}{2T_\mu s(T_\mu s + 1)}.$$

Тогда передаточная функция замкнутого регулятора $F_2(s)$ при охвате отрицательной единичной обратной связью $G_1(s)$ получит вид

$$F_2(s) = \frac{1}{2T_\mu^2 s^2 + 2T_\mu s + 1}.$$

Полученная функция есть замкнутая передаточная функция первого контура, настроенного на модульный оптимум. Переходная функция для $F_2(s)$ представлена на рис. 8.3.

Регулятор второго контура

Структурная схема регулятора величины x_2 представлена на рис. 8.4.

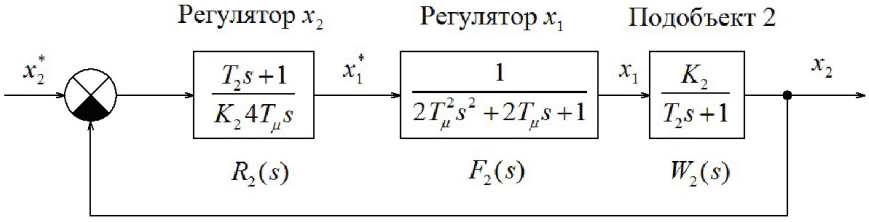


Рис. 8.4. Структурная схема контура регулирования величины x_2

В прямой цепи контура представлены регулятор $R_2(s)$, замкнутый регулятор величины x_1, регулятор $F_2(s)$ и звено подобъекта регулирования с передаточной функцией $W_2(s)$. При синтезе второго регулятора функция $F_2(s)$ играет ту же роль, что и фильтр $F_1(s)$ при синтезе первого регулятора, т. е. $F_2(s)$ является некомпенсируемой частью. Соответственно, передаточная функция регулятора второго контура имеет вид

$$R_2(s)=\left[W_2(s)\right]^{-1}\frac{1}{4T_\mu s}=\frac{T_2 s+1}{K_2}\frac{1}{4T_\mu s},$$

(отличие в том, что вместо $2T_\mu$ во втором контуре берется $4T_\mu$), функция же прямой (разомкнутой) цепи второго регулятора выглядит так:

$$G_2(s)=R_2(s)F_2(s)W_2(s),$$

а после преобразования — так:

$$G_2(s)=\frac{1}{4T_\mu s\left(2T_0^2 s^2+2T_\mu s+1\right)}.$$

Замыкание же $G_2(s)$ единичной отрицательной обратной связью дает типовую передаточную функцию второго контура [8]:

$$F_3(s)=\frac{1}{8T_\mu^3 s^3+8T_\mu^2 s^2+4T_\mu s+1}.$$

Переходная функция для $F_3(s)$ описывается следующим уравнением:

$$x_2(t)=1-e^{-2\tau_2}-\frac{2}{\sqrt{3}}e^{-\tau_2}\sin\left(\sqrt{3}\tau_2\right),$$

где $\tau_2=\frac{t}{4T_\mu}$ — относительное время.

Анализ выражения для $x_2(t)$ показывает, что, как и в рассмотренном ранее случае, вид выражения не зависит от параметров объекта управления, а определяется только величиной нескомпенсированной постоянной времени T_μ.

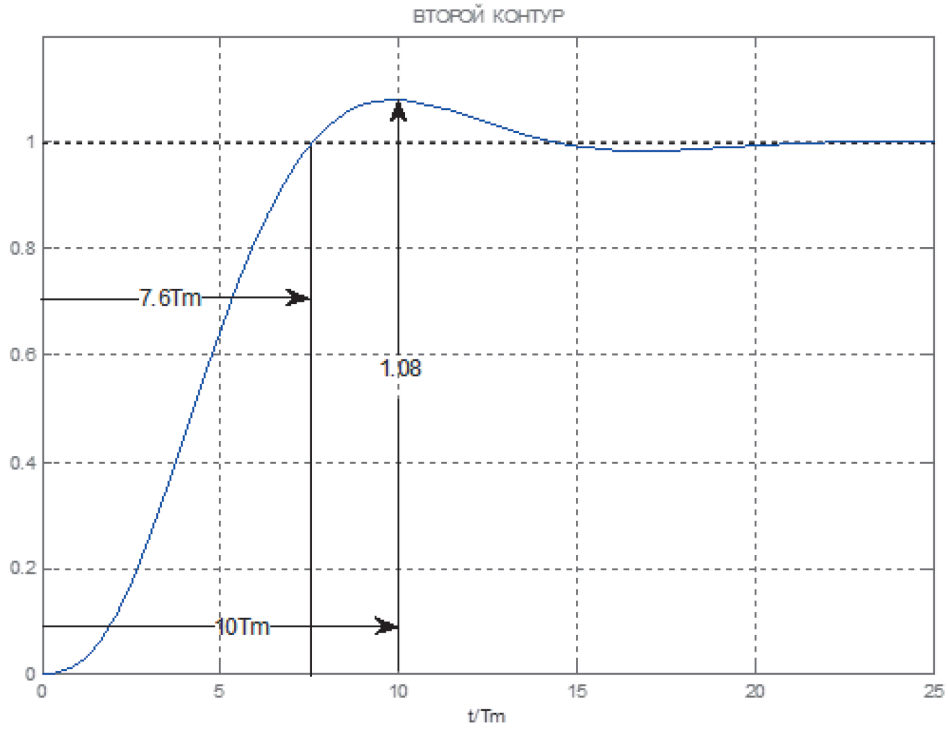


Рис 8.5. Переходная функция регулятора величины x_2

Регулятор третьего контура

Структурная схема регулятора величины x_3 представлена на рис. 8.6.

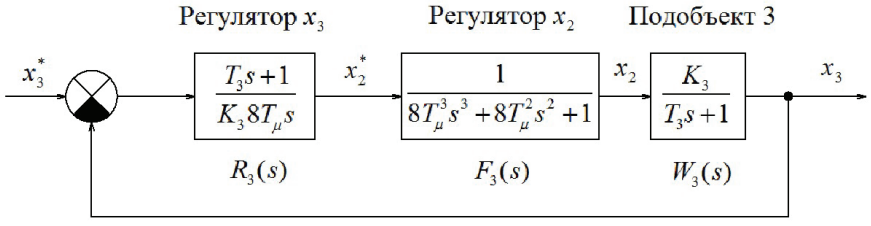


Рис. 8.6. Структурная схема контура регулирования величины x_3

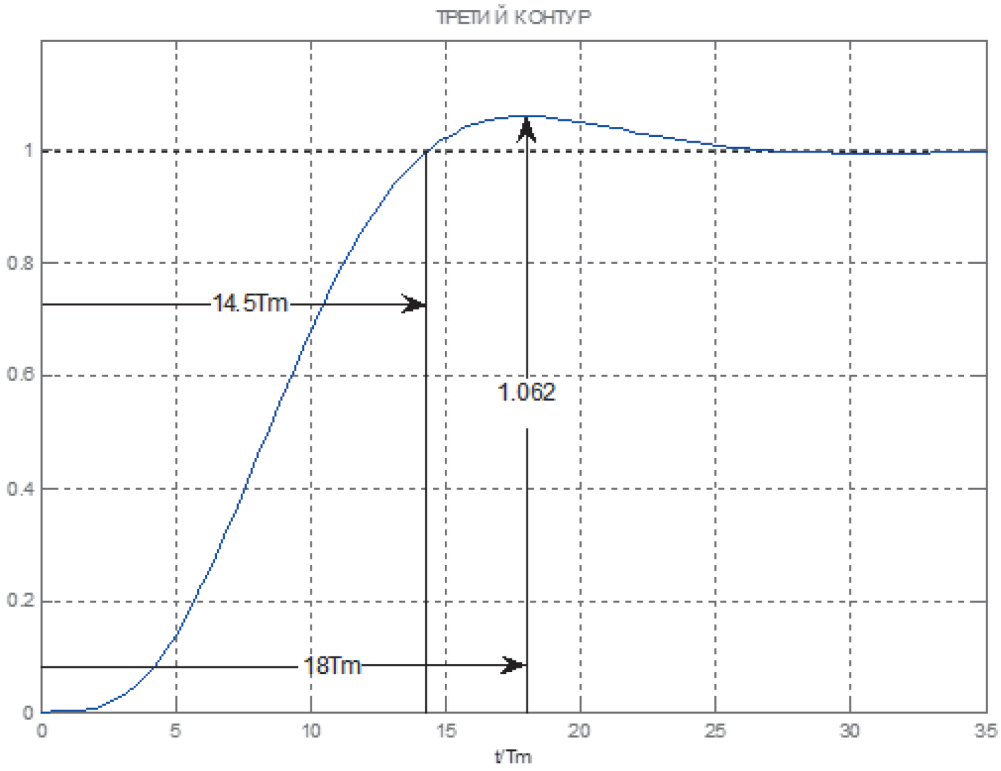


Рис 8.7. Переходная функция регулятора величины x_3

В соответствии с принятой методикой передаточная функция регулятора имеет вид

$$R_3(s) = \left[W_3(s)\right]^{-1} \frac{1}{8T_\mu s} = \frac{T_3 s + 1}{K_3} \frac{1}{8T_\mu s}.$$

Передаточная функция прямой цепи третьего контура имеет вид

$$G_3(s) = R_3(s) F_3(s) W_3(s)$$

или

$$G_3(s) = \frac{1}{8T_\mu s\left(8T_\mu^3 s^3 + 8T_\mu^2 s^2 + 4T_\mu s + 1\right)},$$

а замкнутый единичной отрицательной обратной связью третий контур получает функцию [9]

$$F_4(s) = \frac{1}{64T_\mu^4 s^4 + 64T_\mu^3 s^3 + 32T_\mu^2 s^2 + 8T_\mu s + 1)}.$$

Реакция $F_4(s)$ на единичную функцию при нулевых начальных условиях имеет вид

$$x_3(t) = 1 + e^{-\tau}[(\tau - 1)\cos\tau - (\tau + 2)\sin\tau],$$

где $\tau = \dfrac{t}{4T_\mu}$ — относительное время.

Хорошо заметно, что вид $x_3(t)$ (как и вид $F_4(s)$) не зависит от параметров объекта регулирования, а определяется видом функции $F_4(s)$ и величиной некомпенсируемой постоянной времени T_μ.

8.3. Общие принципы построения и свойства ПСР

Подчинённые системы регулирования с последовательной коррекцией получили широкое распространение в автоматизированных устройствах, применяемых в промышленности, благодаря возможности унифицировать методы синтеза регуляторов, методы наладки автоматизированных систем и устройства, применяемые при реализации систем автоматики.

ПСР характеризуются рядом свойств [8, 10].

1. Регулятор любого контура синтезируется по простой методике, когда его передаточная функция определяется выражением

$$R_i(s) = \left[W_i(s)\right]^{-1} \frac{1}{T_i s},$$

где T_i — постоянная времени, а $R_i(s)$, $W_i(s)$ — регулятор и передаточная функция объекта регулирования, контура с номером i. Важно то, что величина постоянной времени T_i при настройке по модульному (техническому) оптимуму определяется соотношением $T_i = 2^i T_\mu$.

2. ПСР — это совокупность вложенных друг в друга контуров, быстродействие их при типовой настройке уменьшается в два раза при переходе от внутреннего контура к внешнему путем изменения постоянной времени регулятора T_i.

3. Характеристики процессов в системах с ПСР определяются выбором минимальной некомпенсируемой постоянной времени и не зависят от параметров объекта регулирования.

4. Все контуры обеспечивают управление с относительно малым перерегулированием.

Рассмотренные ПСР связаны с передаточными функциями:

$$B_1(s) = \frac{1}{T_\mu s + 1},$$

$$B_2(s) = \frac{1}{2T_\mu^2 s^2 + 2T_\mu s + 1},$$

$$B_3(s) = \frac{1}{8T_\mu^3 s^3 + 8T_\mu^2 s^2 + 4T_\mu s + 1}.$$

Звенья с такими передаточными функциями называются фильтрами Боттерворса первого, второго и третьего порядка. Переходные функции таких фильтров представлены на рис 8.6.

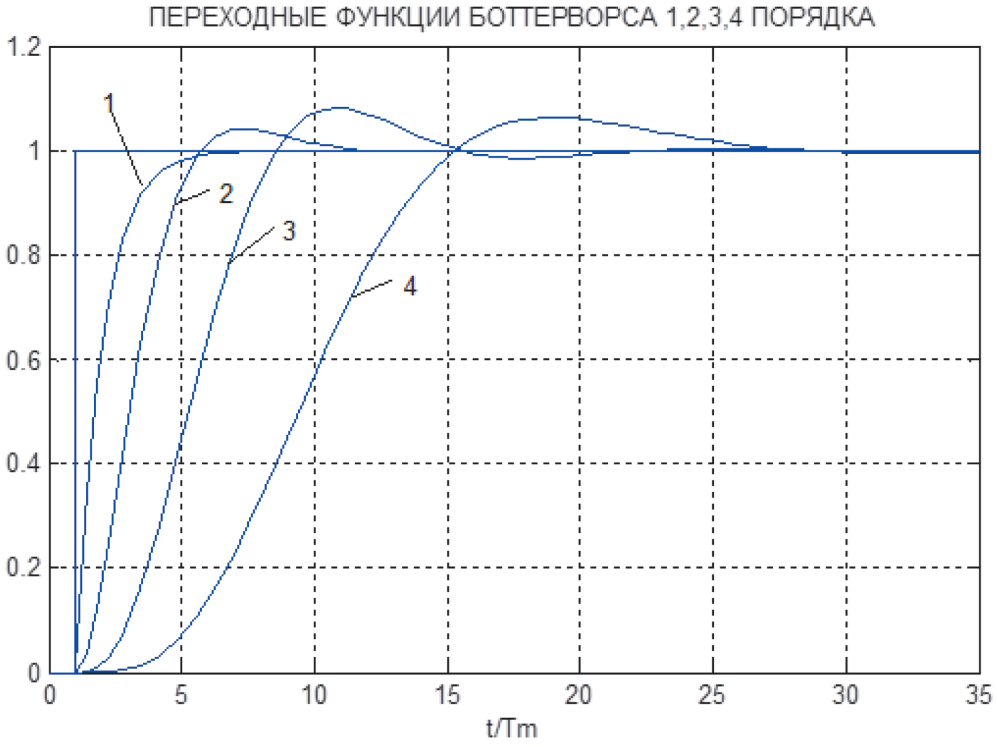


Рис. 8.8 Переходные функции фильтров Боттерворса 1–4 порядка

Можно показать, что звенья с такими передаточными функциями минимизируют функционалы [9]

$$J_1 = \int_0^\infty \left\{ [x_{зд}(t) - x(t)]^2 + T_\mu^2 x'^2(t) \right\} dt,$$

$$J_2 = \int_0^\infty \left\{ [x_{зд}(t) - x(t)]^2 + 4T_\mu^4 x''^2(t) \right\} dt,$$

$$J_3 = \int_0^\infty \left\{ [x_{зд}(t) - x(t)]^2 + 2^6 T_\mu^6 x'''^2(t) \right\} dt,$$

т. е. ограничивается не только квадрат ошибки управления $[x_{зд}(t) - x(t)]^2$, но и производные указанных порядков. Звено четвертого порядка

$$B_4(s) = \frac{1}{64T_\mu^4 s^4 + 64T_\mu^3 s^3 + 32T_\mu^2 s^2 + 8T_\mu s + 1}$$

минимизирует функционал [9]

$$J_4 = \int_0^\infty \left\{ [x_{зд}(t) - x(t)]^2 + 2^7 T_\mu^4 x''^2(t) + 2^{12} T_\mu^8 x''''^2(t) \right\} dt,$$

т. е. ограничивает квадрат второй и четвертой производной, а не только квадрат ошибки. В приведенных формулах величина $[x_{зд}(t) - x(t)]^2$ — квадрат отклонения регулируемой величины от заданной.

Следует добавить, что при типовой переходной функции внешнего контура во внутренних контурах процессы не будут типовыми.

При проектировании регуляторов не следует стремиться реализовывать максимально возможное быстродействие ПСР, поскольку в этом случае силовой элемент (преобразователь, питающий объект) может ухудшить свои энергетические показатели.

Важной особенностью ПСР является возможность ограничивать значения регулируемых переменных за счет применения нелинейных элементов в структуре регуляторов. К примеру, ограничение тока электрического двигателя позволяет защитить двигатель от перегрузки.

Ясно, что условие увеличения постоянной времени регулятора внешнего контура ровно в два раза по отношению к внутреннему контуру не является догмой, и в конкретных условиях возможны какие-то отступления от этого условия.

Заключение

Сегодня MATLAB — это настолько большой по объёму и функциям пакет прикладных программ, что нельзя в небольшом учебном пособии даже в первом приближении охватить все его возможности, поэтому отбор материала всегда даёт читателю повод подвергнуть авторов критике.

Это учебное пособие будет интересно для студентов электротехнических специальностей при первоначальном ознакомлении с матричным калькулятором MATLAB для выполнения контрольных и курсовых работ, а также в студенческой науке, поскольку возможности пакета покрывают весьма значительную часть потребностей студента в вычислительных средствах при изучении самых разных дисциплин, в том числе и математических.

Пособие можно использовать в учёбе как мини-справочник по пакету MATLAB при работе с ним.

Список библиографических ссылок

1. Смирнов Г. Б. Основы управления в среде MATLAB. Режим доступа: http://study.urfu.ru/view/aid_view.aspx?AidId=13593.
2. Дьяконов В. П. MATLAB 6/6.1/6.5 + Simulink 4/5. Основы применения. Полное руководство пользователя. М. : СОЛОН-Пресс, 2004. 472 с.
3. Алексеев Е. Р., Чеснокова О. В. Решение задач вычислительной математики в пакетах Mathcad 12, MATLAB 7, Maple 9. М. : НТ Пресс, 2006. 496 с.
4. Поршнев С. В. MATLAB 7. М. : Бином. Лаборатория знаний, 2006. 320 с.
5. Черных И. В. Моделирование электротехнических устройств в MATLAB, SimPowerSystems и Simulink. 1-е изд. 2007. 288 с.
6. Герман-Галкин С. Линейные электрические цепи: лабораторные работы. СПб. : Корона принт, 2002.
7. Данилов А. Simulink — моделирование в среде Matlab. Компьютерный практикум по курсу «Теория управления». М. : МГУИЭ, 2002.
8. Проектирование электроприводов : справочник / А. М. Вейнгер [и др]. Свердловск : Средне-Уральское кн. изд-во, 1980. 160 с.
9. Техническая кибернетика. Теория автоматического регулирования / под ред. В.В Солодовникова. М. : Машиностроение, 1967. 406 с.
10. Перельмутер В. М., Сидоренко В. А. Системы управления тиристорными электроприводами постоянного тока. М. : Энергоатомиздат, 1988. 304 с.
11. Бородин М. Ю., Метельков В. П., Зеленцов В. И. Расчет систем подчиненного регулирования скорости : методические указания к курсовому проекту «Системы управления электроприводами». Екатеринбург : Изд-во УГТУ-УПИ, 1995. 36 с.

Приложение 1

Избранные матричные операции: определения и специальные символы

Тексты (на английском языке), поясняющие приведённые ниже матричные операции, можно получить на рабочей поверхности MATLAB с помощью команды **help**, например, **help plus** (**help +**) [2, 4, 5].

Арифметические операции

Число арифметических операций в MATLAB значительно расширено и включает в себя матричные и арифметические операции, некоторые из них приведены ниже (A и B — матрицы, x — скаляр).

Команда	Описание	Обозначение (синтаксис)
plus	Плюс	+, (A+B) или plus (A, B)
plus	Плюс	+, (A+x) или plus (A, x)
uplus	Унарный плюс	+, (+A)
minus	Минус	—, (A-B) или minus (A, B)
uminus	Унарный минус	-, (-A)
mtimes	Умножение матриц	*, (A*B) или mtimes (A, B)
mtimes	Умножение матрицы на число	*, (A*x) или mtimes (A, x)
times	Поэлементное перемножение элементов двух матриц	*, (A.*B) или times (A, B)
mpower	Возведение матрицы в степень	^, (A^x) или mpower (A, x)
power	Поэлементное возведение массива в степень	^, (A.^x) или power (A, x)
mldivide	Левое деление матриц (W*X=T, X=W\T)	\, (A\B) или mldivide (A, B)

mrdivide	Правое деление матриц (X*W=T, X=T/W)	/, (A/B) или mrdivide (A, B)
ldivide	Поэлементное деление матриц справа налево	\, (A.\B) или ldivide (A, B)
rdivide	Поэлементное деление матриц слева направо	/, (A./B) или rldivide (A, B)

Операции отношения

Операции отношения имеют результатом логическое значение, служат для сравнения двух величин, векторов или матриц и записываются так, как показано ниже:

Команда	Описание	Обозначение (синтаксис)
eq	Эквивалентно	==, (X==Y) или eq (X, Y)
ne	Не эквивалентно	~=, (X~=Y) или ne (X, Y)
lt	Меньше	<, (X <Y) или lt (X, Y)
gt	Больше	>, (X >Y) или gt (X, Y)
le	Меньше или равно	<, (X<=Y) или le (X, Y)
ge	Больше или равно	>=, (X >=Y) или ge (X, Y)

X и Y могут быть или матрицами одинаковой размерности, или скалярами. В случае матриц речь идёт о поэлементных операциях.

Логические операции

Рассмотренные ниже операторы поэлементно реализуют логические операции над массивами одинаковой размерности.

Команда	Описание	Обозначение (синтаксис)
and	Логическое И	&, A & B или and (A, B)
or	Логическое ИЛИ	, A | B или or (A, B)
not	Логическое НЕ	~, ~ A или not (A)
xor	Исключающее ИЛИ	xor (A, B)

Таблица истинности для этих операторов и функций имеет следующий вид:

Переменные		not (A)	and (A, B)	or (A, B)	xor (A, B)
A	B	~A	A&B	A|B	
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

При обработке массивов эти операции применяются поэлементно.

Приложение 2

Основные функции и команды MATLAB из теории управления

Ниже приведён перечень команд MATLAB из теории управления с описанием.

Команда	Описание
>>команда …;	Завершение команды; блокирует выдачу на экран результата или содержания переменной
>>help команда	Выдача краткого описания команды или функции на экран на английском языке
>>who	Распечатывает текущие переменные и параметры рабочей области
>>whos	Дополнительно выдаёт имя, тип, размерность и тип матрицы (плотная, редкая и т. д.)
>>diary<'имя файла'>	Включает режим записи выполненных команд в файл, а также результат их выполнения
>>diary on **>>diary off**	Включают и выключают режим записи (**>>diary** переключает режим on на off и наоборот)
%	На рабочей поверхности MATLAB или в М-файле указывает на строку комментария
>>format compact **>>format long** **>>format short** …	Основная задача команды — это определение формата представления используемых в вычислениях чисел (длина от 3 до 15 десятичных знаков). Сами вычисления всегда выполняются с двойной точностью
>>save<имя файла> <переменные>	Выгружает переменные из рабочей области в файл, при указании имён переменных выгружаются только указанные переменные (возможны опции)
>>load<'имя файла'>	Считывает данные из файла

Команда	Описание
>>clear all **>>clear <имя1>,<имя2>,..**	Команда удаляет все переменные из рабочей области или только указанные
>clc	Очищает командное окно, но переменные с рабочей поверхности не удаляет
>>echo on **>>echo off**	Включает и выключает вывод на экран содержимого Script-файла
>>echo <'имя файла'>on **>>echo <'имя файла'>off**	Разрешает (отменяет) вывод на экран текста М-файла
>>V=A(:,3)	Задание вектора V, состоящего из элементов третьего столбца матрицы A
>>V=A(2,:)	Задание вектора V, состоящего из элементов второй строки матрицы
>>B=A(2:3,2:4)	Задание матрицы B, состоящей из пересечения элементов второй и третьей строки и элементов второго, третьего и четвёртого столбцов матрицы A
>>G_s=tf(num, den) **>> G_s=tf([a_n.. a_2 a_1 a_0], [b_m.. b_2 b_1 b_0..])**	Задание передаточной функции $G_s(s)=\frac{a_n s^n+...+a_1 s+a_0}{b_m s^m+...+b_2 s^2+b_1 s+b_0}$, где num –числитель, a den — знаменатель, либо представленные как вектор-строки и определённые заранее, либо определённые явно в виде соответствующих векторов. При равенстве нулю всех коэффициентов числителя, кроме a_0, угловые скобки числителя могут быть опущены
>>conv([a_n.. a_2 a_1 a_0], [b_m.. b_2 b_1 b_0.])	Перемножение двух полиномов, коэффициенты которых заданы в виде векторов-строк. Например, команда G_s = tf(5, conv([1 2], [3 4])) даёт следующий результат: Transfer function: 5 ———————————— s^2+10s+8
>>zpk(G_s)	Представляет числитель и знаменатель передаточной функции в виде элементарных сомножителей с действительными коэффициентами, после чего передаточную функцию можно представить в виде комбинации основных динамических звеньев

Команда	Описание
>>zpk(G$_s$)	Пусть передаточная функция задана в виде $G_s = \frac{s^2+11s+30}{s^3+9s^2+28s+30}$, тогда применение команды даст следующий результат: Zero/pole/gain: $\frac{(s+6)\ (s+5)}{(s+3)\ (s^2+6s+10)}$.
Gs=parallel(G1, G2) **Gs=parallel(Gs, G2)** **Gs=G1+G2+G3**	Команда **parallel** даёт передаточную функцию G_s, эквивалентную двум включённым параллельно передаточным функциям G1 и G2. Однократное применение команды **parallel** даёт возможность запараллелить только две функции
Gs=series(G1, G2,..) **Gs=G1*G2***…	Команда **series** даёт передаточную функцию G_s, эквивалентную включённым последовательно передаточным функциям G1, G2, ...
pole(Gs)	Команда выдаёт полюса (в том числе комплексно-сопряжённые) для передаточной функции Gs в виде вектора-столбца
tzero(Gs)	Команда выдаёт нули (в том числе комплексно-сопряжённые) для передаточной функции G_s в виде вектора-столбца
abs(Var)	Команда выдаёт (в том числе для комплексного числа) абсолютное значение переменной. При переменной матрице вычисляются модули её элементов
pzmap(Gs)	Команда представляет на комплексной плоскости нули (в виде 0) и полюса (в виде x) для передаточной функции G_s
plot(x, y) **plot(x1, y1, x2, y2,..)**	Команда рисует графики функций (x-, y-массивы соответствующих значений аргумента и функции). Возможно представление нескольких графиков с помощью одной команды
subplot(m, n, z) **subplot(3,1,1); step(Gs);** **subplot(3,1,2); bode(Gs);**	В одном окне команда резервирует место для выдачи нескольких графиков, размещение которых упорядочено по строкам (m) и столбцам (n). Графики нумеруются в порядке слева направо и сверху вниз. Переменная z указывает место размещения соответствующего графика

Команда	Описание

`step(Gs);`
`step(Gs, tн: dt: tк);`
`step(Gs1, Gs2,..);`
	Команда обеспечивает графическое представление реакции звена на ступенчатое воздействие, где $t_н$, $t_к$ — начальное и конечное время переходного процесса, а dt — шаг. На одном графике можно представить реакцию нескольких звеньев. Первая команда обеспечивает начало процесса с $t_н = 0$

`figure`	Команда открывает новое графическое окно, оставляя предыдущие

`hold`
`hold on`
`hold off`
	Команда **hold** или **hold on** сохраняет все актуальные графики с заданными масштабами осей, легендами, надписями, решетками и прочими установками. Последующие кривые, задаваемые командой **plot** добавляются к существующим. В дальнейшем командой **hold** или **hold off** отменяется режим продолжения графических построений

`grid`	Для лучшего обозрения вставляется сетка. Последовательное применение команды включает и выключает выдачу сетки
`feedback(Go, G1)` `feedback(Go,1)`	Команда вычисляет передаточную функцию для G_0, охваченной отрицательной обратной связью, в которую включена функция G1. Во втором случае обратная связь единичная
`w=logspase(d1, d2, n)`	Команда формирует вектор-строку, содержащую n десятичных логарифмов чисел, распределённых в границах от 10^{d1} до $10^{d2.}$ Если величина n не задана, то считается, что n = 50, d ∈ [d1, d2] — равномерно распределённые на отрезке числа. Для ω от ω = 0,1 до ω = 100, w = logspase (−1,2)
`w=linspase(x1, x2, n)`	Команда похожа на предыдущую, но здесь x1 и x2 — начальное и конечное значения множества n чисел, равномерно распределённых на отрезке [x1, x2]
`[p]=rlocus(Gs, k)` `[p, k]=rlocus(Gs)`	Команда **[p]=rlocus(Gs, k)** присваивает вектору p значения полюсов (в комплексной форме) для коэффициента усиления k в методе корневого годографа. Команда **[p, k]=rlocus(Gs)** выдаёт значения полюсов для наперёд заданного множества значений переменной k

Команда	Описание

`rlocus(Go, k)`
`rlocus(Go, k`$_н$`: dk: k`$_к$`)`
`rlocus(Go)`
	Команда **`rlocus(Go, k)`** выдаёт годограф корней для заранее определённого вектора коэффициентов усиления k, причём нули и полюса маркируются. Для k = 1 получается такой же результат, как и для команды **`pzmap(Go)`**. Можно также задавать область изменений коэффициента k с начальным и конечным значениями: $k_н$, $k_к$. Если коэффициент не задан, то корневой годограф строится для области значений $k \in [0, \infty]$, причём полюс маркируется только для k = 1

`[k, p]=rlocfind (Gs)`	Команда представляет годограф нулей и полюсов для области значений k. Выбрав положение полюса с помощью креста, а затем щёлкнув по нему, можно получить численные значения k и полюса

`nyquist(Gs)`
`nyquist(Gs, w)`
`[re, im, w]=nyquist(Gs)`
`[re, im]=nyquist(Gs, w)`
	Команда **`nyquist(Gs)`** выдаёт годограф кривой Найквиста. Команда **`[re, im, w]=nyquist(Gs)`** определяет величины действительной и мнимой частей вектора для текущего значения частоты w. Область частот заранее задаётся, например, в команде **`[re, im]=nyquist(Gs, w)`**. Щелчок правой клавишей мыши на поверхности рядом с кривой Найквиста окрывает окно, в котором через опцию **`Show`** можно деактивировать нежелательное представление кривой в области отрицательных частот

`bode(Gs)`
`bode(Gs, w)`
`[a, p, w]=bode(Gs)`
`[a, p]=bode(Gs, w)`
	Команда **`bode(Gs)`** выдаёт графики амплитудно-частотной и фазо-частотной характеристик (диаграмму Боде). Команда **`[a, p, w]=bode(Gs)`** присваивает значения амплитуды вектору a и значения фазы вектору p для текущих значений частоты. Область частот w предварительно задаётся командой **`bode(Gs, w)`**, например так: **`[a, p]=bode(Gs, w)`**

`margin(Gs)`	Команда представляет на диаграмме Боде запас по фазе и амплитуде, по фазе — при значении амплитуды равном 1, а по амплитуде, соответственно, при значении фазы −180°. Запасы по амплитуде значением −0 dB и фазе со значением −0° говорят о том, что замкнутый регулятор находится на границе устойчивости

Оглавление

Учебное издание

Смирнов Геннадий Борисович
Томашевич Виктор Григорьевич

ЛИНЕЙНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ В ПАКЕТЕ MATLAB

Редактор М. А. Терновая
Вёрстка О. П. Игнатьевой

Подписано в печать 17.04.2018. Формат 70×100/16.
Бумага офсетная. Цифровая печать. Усл. печ. л. 6,1.
Уч.-изд. л. 3,5. Тираж 50 экз. Заказ 49.

Издательство Уральского университета
Редакционно-издательский отдел ИПЦ УрФУ
620049, Екатеринбург, ул. С. Ковалевской, 5
Тел.: +7 (343) 375-48-25, 375-46-85, 374-19-41
E-mail: rio@urfu.ru

Отпечатано в Издательско-полиграфическом центре УрФУ
620083, Екатеринбург, ул. Тургенева, 4
Тел.: +7 (343) 358-93-06, 350-58-20, 350-90-13
Факс: +7 (343) 358-93-06
http://print.urfu.ru


УНИВЕРСИТЕТСКОГО КНИГОИЗДАНИЯ
УНИВЕРСИТЕТСКАЯ КНИГА
2015


9 785799 623852