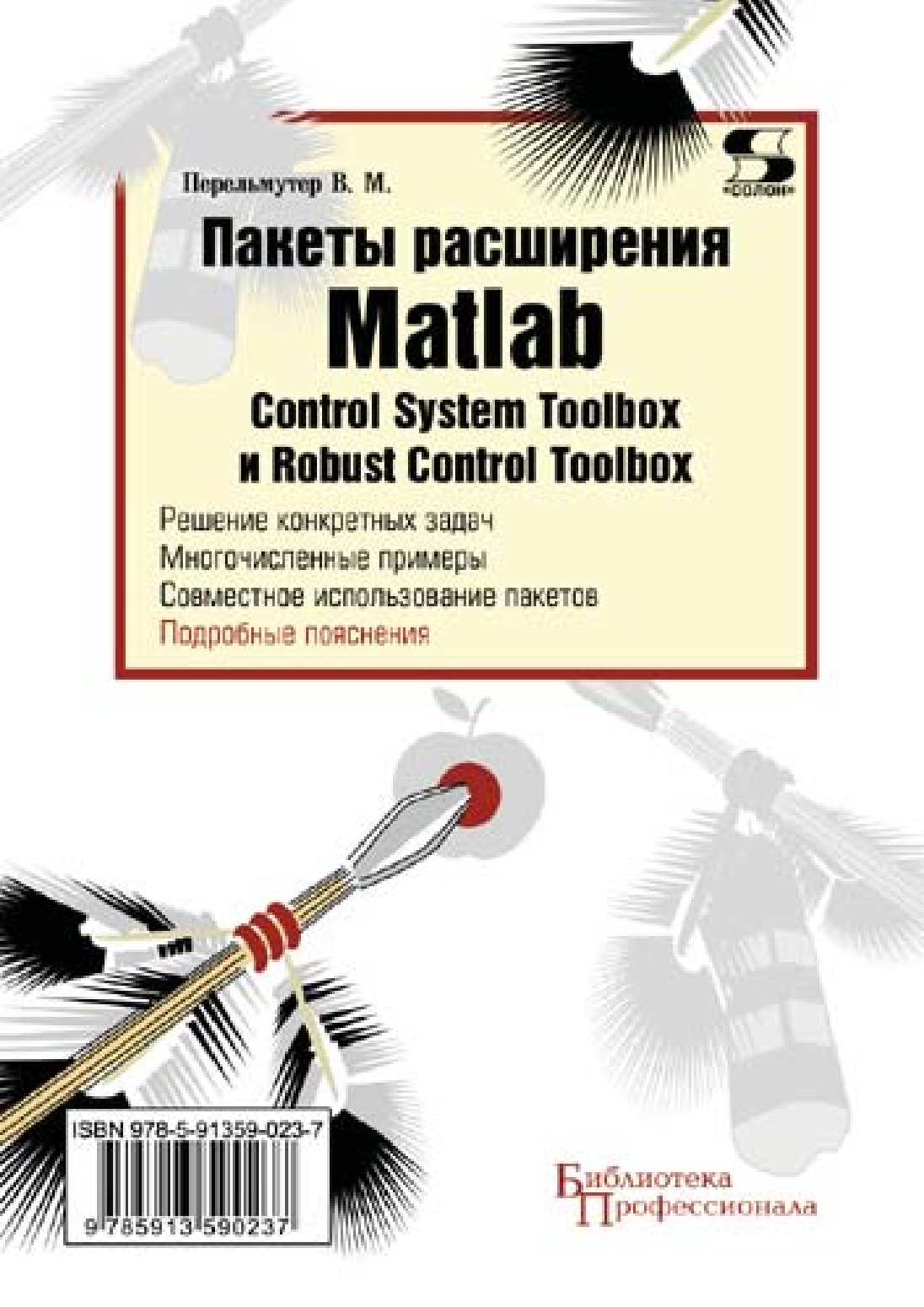
Перельмутер В. М.

«СОЛОН»

Пакеты расширения Matlab

Control System Toolbox и Robust Control Toolbox

Решение конкретных задач
Многочисленные примеры
Совместное использование пакетов
Подробные пояснения



ISBN 978-5-91359-023-7

Библиотека Профессионала

УДК 681.3
ББК 32.884.1
П27

В. М. Перельмутер

П27 Пакеты расширения MATLAB. Control System Toolbox и Robust Control Toolbox — М.: СОЛОН-ПРЕСС, 2009. — 224 с.: ил. — (Серия «Библиотека профессионала»).

ISBN 978-5-91359-023-7

В книге описаны пакеты расширения Control System Toolbox и Robust Control Toolbox системы MATLAB 7, предназначенные для анализа и синтеза систем управления. Коротко изложены основные теоретические положения, принятые при разработке указанных пакетов расширения. Приведены многочисленные примеры использования этих пакетов расширения для решения конкретных задач с подробным пояснением выполняемых операций. Показаны возможности совместного использования пакетов Control System Toolbox и Robust Control Toolbox с пакетом Simulink, что увеличивает возможности всех трех указанных пакетов расширения.

Книга может быть использована студентами вузов соответствующих специальностей при курсовом и дипломном проектировании, инженерами и научными работниками при создании новых и исследованиях уже разработанных систем автоматического управления. Книга рассчитана как на начинающих, так и на достаточно опытных пользователей

По вопросам приобретения обращаться: ООО «АЛЬЯНС-КНИГА КТК»
Тел: (495) 258-91-94, 258-91-95 **www.abook.ru**
Сайт издательства «СОЛОН-ПРЕСС» **www.solon-press.ru**.
E-mail: **solon-avtor@coba.ru**

ISBN 978-5-91359-023-7

Глава 1
Способы описания линейных динамических систем

1.1. Методы фазовых координат (пространственных состояний)

В пакетах расширения Control System Toolbox, Robust Control Toolbox системы MATLAB приняты следующие способы описания линейных динамических систем с постоянными параметрами: система уравнений первого порядка в фазовом пространстве, или в пространстве состояний системы (*SS* — state-space), передаточная функция системы в виде отношения двух полиномов (*TF*), передаточная функция в так называемом виде нуль/полюс/ коэффициент усиления (*ZPK*). Наибольшее распространение получил первый способ, который дает наилучшую точность при вычислениях и в большей степени удобен при теоретических исследованиях и практической реализации алгоритмов управления с применением вычислительных машин и т. д.

При применении этого способа дифференциальные уравнения, описывающие динамику системы, имеют вид:

$$\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \qquad (1.1)$$

где $\mathbf{x}$ — n-мерный вектор состояния системы (вектор фазовых координат), $\mathbf{u}$ — p-мерный вектор внешних воздействий, состоящий из заданных величин, возмущений, управлений, формируемых регулятором, $\mathbf{A}$ и $\mathbf{B}$ — переходная матрица системы и матрица управлений соответствующих размеров. Предполагается, что измерению доступна только часть состояний системы или их линейных комбинаций, такие переменные $\mathbf{y}$ называются выходами системы:

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \qquad (1.2)$$

где **y** — m-мерный выходной вектор, **C, D** — матрицы соответствующих размеров. Для большинства реальных объектов управления **D** = 0.

В рассматриваемых пакетах расширения имеется возможность манипулировать с системой, описываемой уравнениями (1.1), (1.2), как с одним объектом MATLAB. Для этого нужно матрицы **A, B, C, D** трансформировать в систему, используя команды **ss** из Control System Toolbox или **mksys** из Robust Control Toolbox.

Далее в книге в качестве примера часто будем рассматривать вращающуюся механическую систему, состоящую из двух инерционных масс J_1, J_2, соединенных упругой муфтой (рис. 1.1).

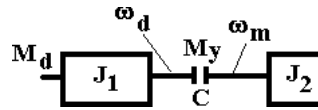


Рис. 1.1. Механическая система с эластичной муфтой

Момент M_y, передаваемый муфтой, который является моментом сопротивления для первой инерционной массы и движущим моментом для второй, пропорционален разности угловых перемещений обеих масс с коэффициентом пропорциональности C. Управлением является приводной момент первой массы (обычно момент приводного двигателя) M_d, а выходом — скорость ее вращения ω_d, так как на валу приводного двигателя обычно устанавливается датчик скорости. Для простоты момент сопротивления полагаем равным нулю. Уравнения системы:

$$\frac{d\omega_d}{dt} = (M_d - M_y)/J_1, \tag{1.3}$$

$$\frac{dM_y}{dt} = C(\omega_d - \omega_m) \tag{1.4}$$

$$\frac{d\omega_m}{dt} = M_y / J_2 \tag{1.5}$$

Если обозначить $\mathbf{x} = (\omega_d, M_y, \omega_m)^T$, $u = M_d$, то рассматриваемую систему можно записать в виде (1.1), (1.2) с матрицами:

$$\mathbf{A} = \begin{bmatrix} 0 & -1/J_1 & 0 \\ C & 0 & -C \\ 0 & 1/J_2 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1/J_1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{C} = [1\ 0\ 0], \quad \mathbf{D} = 0 \tag{1.6}$$

Теперь создадим систему, используя команды **ss** или **mksys**. Создадим *m*-файл (П. 1.1). Параметры примера взяты для вращающейся печи цементного производства. В командном окне MATLAB откроем последовательно окна File, New, *M*-File и введем текст (конечно, комментарий можно не вводить):

```
clear %очистка рабочей области памяти
clc   %очистка командного окна
J1 = 21.5;J2 = 7;C = 243; %ввод параметров
a = [0  –1/J1  0; C  0  -C; 0  1/J2  0]; %создание матрицы A
b = [1/J1; 0; 0]; c = [1  0  0]; d = 0;  %создание матриц B, C, D
sys1 = ss(a, b, c, d);
%формирование системы для Control System Toolbox
sys2 = mksys(a, b, c, d);
%формирование системы для Robust Control Toolbox.        (П. 1.1)
```

Отметим следующее: Матрицы можно вводить как одну текстовую строку, причем строки матрицы отделяются друг от друга точкой с запятой, а отдельные элементы строки матрицы разделяются пробелами. Если требуемая длина текстовой строки превышает ширину страницы, то можно переносить ее на следующую строку, причем в месте разрыва должны быть поставлены точки, не менее трех. Если после команды имеется точка с запятой, то результаты выполнения команды в командном окне не появляются. Если выполнить эту программу (в окне редактора выполнить Debug и Run или Save and Run), то результат запоминается в рабочей области (Workspace). Если теперь открыть ее (в окне MATLAB выполнить Desktop, затем Workspace), то появится содержание рабочей области с двумя среди прочих объектами sys1, sys2. Если теперь раскрыть эти системы путем двойного нажатия на их символы левой клавиши мыши, то будет видно, что эти системы организованы совершенно по-разному: первая представляет собой собрание матриц **a, b, c, d**, а вторая — столбец с 51 числом. Во многих случаях команды из Robust Control Toolbox можно использовать с объектами, образованными командой **ss,** но системы, образованные командой **mksys,** обычно не работают с командами из Control System Toolbox.

При рассмотрении sys1 видно, что переменные имеют типовые обозначения $x1$,$x2$ и т. д., часто желательно иметь имена, отражающие физическую суть величин. Добавим команду

```
set(sys1,'StateName',{'Wd'; 'My'; 'Wm'},'InputName','Md',...
'OutputName','Wd')                                    (П. 1.1.1)
```

и выполним программу, раскроем рабочую область, раскроем sys1 и увидим, что теперь переменные имеют нужные имена. Отметим, что такой же результат можно получить, если приведенную строку добавить в команду **ss**.

При некоторых операциях в Robust Control Toolbox формируется (вычисляется) искомая система в форме, принятой в этом пакете расширения, а интерес будет представлять ее описание в виде (1.1), (1.2) или же отдельные матрицы этой системы. При этом используется команда **branch**. Если, например, в (П. 1.1) ввести команду

[aq,bq]=branch(sys2), (П. 1.1.2)

то в рабочей области и в командном окне получим матрицы **aq** = **a**, **bq** = **b**.

Уже в виде (1.1), (1.2) нас может интересовать ряд характеристик системы. В первую очередь, является ли система устойчивой и как располагаются ее полюса, т. е. корни характеристического уравнения. Воспользуемся командой

Pa = eig(a), (П. 1.1.3)

получим 0; 0 + 6.7836i; 0 – 6.7836i, т. е. все три корня расположены на мнимой оси (i — знак мнимой единицы).

Для того, чтобы системой можно было нормально управлять, она должна, как правило, быть управляемой и наблюдаемой. Система является полностью управляемой, если ее можно из любого начального состояния привести в нулевое состояние за конечное время, используя произвольно пространство управлений. Пусть, например, в системе одно из уравнений имеет вид $dx_i/dt = ax_i$, тогда очевидно, что на процесс изменения x_i влиять нельзя, т. е. система является неуправляемой. Система является полностью наблюдаемой, если по наблюдениям в течение конечного времени за ее выходом можно определить начальное состояние системы. Пусть в системе третьего порядка второе уравнение имеет вид: $dx_2/dt = -ax_3 + bu$, причем координата x_2 не входит ни в уравнения для x_1, x_3, ни в выражение для y. Таким образом, координата x_2 не «проявляет» себя на выходе системы ни непосредственно, ни через другие координаты, и система оказывается ненаблюдаемой.

Для управляемости системы требуется, чтобы матрица управляемости

$$\mathbf{C}_0 = (\mathbf{B}, \mathbf{AB}, \mathbf{A}^2\mathbf{B}, \mathbf{A}^{n-1}\mathbf{B}) \qquad (1.7)$$

имела ранг *n*. Напомним, что ранг матрицы равен порядку наибольшего определителя матрицы, отличного от нуля, при условии, что все старшие определители равны нулю. Соответственно для наблюдаемости системы требуется, чтобы матрица наблюдаемости

$$\mathbf{D}_0 = (\mathbf{C},\ \mathbf{CA},\ \mathbf{CA}^2,\ \mathbf{CA}^{n-1})^T \tag{1.8}$$

имела ранг *n*. Для проверки управляемости и наблюдаемости нашей системы добавим в (П. 1.1) команды

```
Co = ctrb(a,b);
Unco = length(a)-rank(Co)
Do = obsv(sys1);
Undo = length(a)-rank(Do)
```
(П. 1.1.4)

Первая и третья команды формируют матрицы управляемости и наблюдаемости, а вторая и четвертая вычисляют разности между рангом этих матриц и порядком системы. После выполнения нашей программы получим в командном окне *Unco* = 0, *Undo* = 0, т. е. наша система полностью управляема и наблюдаема.

Обратим внимание, что аргументом этих команд могут быть как входящие в искомые матрицы системные матрицы (первая команда), так и сама система (третья команда). С системой sys2 эти команды не выполняются.

Рассматриваемые системы в общем случае имеют несколько входов и выходов и относятся к так называемым MIMO системам (multiple-input — multiple-output). Частным, но достаточно распространенным на практике является случай систем с одним входом и одним выходом (SISO системы). Методы анализа и синтеза MIMO систем могут быть применены к SISO системам, однако последние более простые, и для работы с ними могут быть использованы специальные методы.

Для системы (1.1), (1.2) иногда используется обозначение

$$sys = \left[\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \mathbf{C} & \mathbf{D} \end{array}\right]. \tag{1.9}$$

Это значит, что система sys описывается уравнениями (1.1), (1.2). Часто также в системе входные и выходные переменные разделяются: входы — на внешние воздействия **w** и регулирующие сигналы **u**, выходы — на регулируемые $\mathbf{y}_1$ и измеряемые $\mathbf{y}_2$. При этом уравнения системы приобретают вид:

$$\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x} + \mathbf{B}_1\mathbf{w} + \mathbf{B}_2\mathbf{u} \tag{1.10}$$

$$\mathbf{y}_1 = \mathbf{C}_1\mathbf{x} + \mathbf{D}_{11}\mathbf{w} + \mathbf{D}_{12}\mathbf{u}, \tag{1.11}$$

$$\mathbf{y}_2 = \mathbf{C}_2\mathbf{x} + \mathbf{D}_{21}\mathbf{w} + \mathbf{D}_{22}\mathbf{u}. \tag{1.12}$$

Система в таком виде обозначается как

$$sys = \left[\begin{array}{c|cc} \mathbf{A} & \mathbf{B}_1 & \mathbf{B}_2 \\ \hline \mathbf{C}_1 & \mathbf{D}_{11} & \mathbf{D}_{12} \\ \mathbf{C}_2 & \mathbf{D}_{21} & \mathbf{D}_{22} \end{array}\right]. \tag{1.13}$$

Предполагается, что регулятор на основании измерений $\mathbf{y}_2$ вырабатывает сигнал $\mathbf{u}$ таким образом, что выход $\mathbf{y}_1$ при заданном $\mathbf{w}$ имеет желаемые свойства (рис. 1.2).

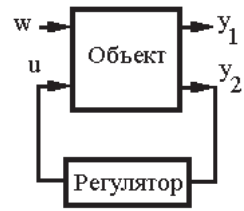


Рис. 1.2. Структурная схема системы с разделенными входами и выходами

Учтем в рассматриваемой системе (1.3) — (1.6) момент нагрузки M_c и предположим, что цель регулирования — уменьшить отклонения скорости механизма ω_m при действии M_c. Запишем (1.5) как

$$\frac{d\omega_m}{dt} = M_y / J_2 - M_c / J_2 \tag{1.14}$$

и обозначим $w = M_c$, $y_1 = \omega_m$, $y_2 = \omega_d$,

$$\mathbf{B}_1 = \begin{bmatrix} 0 \\ 0 \\ -1/J_2 \end{bmatrix}, \quad \mathbf{B}_2 = \begin{bmatrix} 1/J_1 \\ 0 \\ 0 \end{bmatrix}, \quad C_1 = [0 \;\; 0 \;\; 1], \quad C_2 = [1 \;\; 0 \;\; 0] \tag{1.15}$$

Получаем систему в виде (1.13) при всех $\mathbf{D}_{ij} = 0$.

В исследованиях линейных систем с постоянными параметрами достаточно часто используются так называемые канонические формы дифференциальных уравнений. Произведем в (1.1) замену переменных по формуле $\mathbf{z} = \mathbf{T}\mathbf{x}$, где $\mathbf{T}$ — неособая матрица, получим

$$\mathbf{T}^{-1}\frac{d\mathbf{z}}{dt} = \mathbf{A}\mathbf{T}^{-1}\mathbf{z} + \mathbf{B}\mathbf{u} \tag{1.16}$$

или

$$\frac{d\mathbf{z}}{dt} = \mathbf{A}_z\mathbf{z} + \mathbf{B}_z\mathbf{u} \tag{1.17}$$

$$\mathbf{y} = \mathbf{C}_z\mathbf{z} + \mathbf{D}\mathbf{u} \tag{1.18}$$

$$\mathbf{A}_z = \mathbf{T}\mathbf{A}\mathbf{T}^{-1}, \quad \mathbf{B}_z = \mathbf{T}\mathbf{B}, \quad \mathbf{C}_z = \mathbf{C}\mathbf{T}^{-1}.$$

Матрица $\mathbf{T}$ выбирается таким образом, чтобы получить одну из двух канонических форм для системы (1.17), (1.18). В канонической форме «modal» матрица $\mathbf{A}_z$ имеет диагональный вид, причем на главной диагонали расположены вещественные корни характеристического уравнения, а каждая пара комплексных корней образует блок 2*2 на главной диагонали. Если, например, система 4-го порядка имеет корни α, $\gamma + j\omega$, $\gamma - j\omega$, β, то

$$\mathbf{A}_z = \begin{bmatrix} \alpha & 0 & 0 & 0 \\ 0 & \gamma & \omega & 0 \\ 0 & -\omega & \gamma & 0 \\ 0 & 0 & 0 & \beta \end{bmatrix} \tag{1.19}$$

В канонической форме «companion» матрица $\mathbf{A}_z$ имеет следующий вид:

$$\mathbf{A}_z = \begin{bmatrix} 0 & 0 & .. & .. & .. & -a_n \\ 1 & 0 & 0 & .. & .. & -a_{n-1} \\ 0 & 1 & 0 & .. & .. & -a_{n-2} \\ 0 & 0 & 1 & .. & .. & .. \\ 0 & 0 & 0 & 1 & 0 & -a_2 \\ 0 & 0 & 0 & 0 & 1 & -a_1 \end{bmatrix}, \tag{1.20}$$

где $a_1 ... a_n$ — коэффициенты характеристического уравнения системы

$$p(s) = s^n + a_1 s^{n-1} + ... + a_{n-1}s + a_n. \tag{1.21}$$

Дополним нашу программу П. 1.1 командами

```
Msys11=canon(sys1, 'modal')
Msys12=canon(sys1, 'companion')
```
(П. 1.1.5)

и выполним ее. Получим для первой команды

a = [0 0 0;0 0 6.784;0 -6.784 0]

и для второй

a = [0 0 0;1 0 –46.02;0 1 0].

Если учесть приведенные выше значения корней характеристического полинома и что в этой связи этот полином записывается как $(s - 6.784j)(s + 6.784j)\mathrm{s} = s^3 + 46.02s$, то видно, что формулы (1.19), (1.20) действительно реализуются.

Приведенное **T**-преобразование используется также для балансировки системных матриц. Дело в том, что довольно часто образующие эти матрицы элементы отличаются друг от друга на несколько порядков. При этом выполняемые компьютером вычисления могут оказаться не полностью достоверными из-за конечной длины представления чисел в нем. Операция балансировки заменяет фактические системные матрицы более пригодными для численных расчетов, уравнивая различные нормы матрицы **A**. Существует несколько норм матрицы. В частности, норма матрицы по строкам определяется как наибольшая из сумм абсолютных значений чисел каждой строки, а норма матрицы по столбцам определяется как наибольшая из сумм абсолютных значений чисел каждого столбца. Операция балансирования делает эти две нормы примерно равными. Для целей балансирования используется команда

[sysb,T] = ssbal(sys).

Команда возвращает вместо исходной системы сбалансированную, а также матрицу преобразования **T**, позволяющую при необходимости рассчитать фактические фазовые координаты системы.

Рассмотрим пример из [Л.1]. Пусть

$$\mathbf{A} = \begin{bmatrix} 1 & 10^4 & 10^2 \\ 0 & 10^2 & 10^5 \\ 10 & 1 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad \mathbf{C} = [0.1 \;\; 10 \;\; 100].$$

Создадим *m*-файл, содержащий данные этих матриц, дополним его командами

```
sys = ss(A,B,C,0)
[sysb,T] = ssbal(sys),
```

а затем выполним его. В результате получим

```
a = [1  2500  0.3906;0  100  1563;2560  64   0],
b = [0.125;0.5;32],
c = [0.8   20 3.125],
T = [0.1250  0  0;0  0.5000  0;0   0  32.0].
```

Видно, что новая матрица **a** хорошо сбалансирована.

В Control System Toolbox возможно также использовать описание систем в виде уравнений состояний с дескриптором. Так называется система уравнений в виде

$$\mathbf{E}\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \qquad (1.22)$$

Матрица **E** часто получается естественным путем при записи уравнений состояний системы. Для получения уравнений в форме (1.1) ее надо обратить (см. (1.16), (1.17)). Если **E** плохо обусловлена, то при последующих операциях с системой возможны ошибки, поэтому работа с системой в форме (1.22) будет предпочтительнее. Формирование системы с дескриптором производится командой

```
sys = dss(A, B, C, D, E).
```

Принятое в Control System Toolbox описание динамических систем в фазовом пространстве допускает наличие чистого запаздывания на входах и выходах системы. Пусть запаздывание на входах равно 0.1 с, а на выходах 0.2 с. Дополним программу (П. 1.1) командами

```
sys1T=ss(a, b , c, 0, 'inputdelay',0.1, 'outputdelay',0.2)
impulse(sys1)
hold on
impulse(sys1T).
```
(П. 1.1.6)

Первая команда формирует систему с указанным запаздыванием, а остальные рассчитывают и рисуют на одном и том же графике процессы в системе без запаздывания и с запаздыванием при импульсном воздействии (рис. 1.3). Видно запаздывание колебаний второй системы относительно первой. Если раскрыть рабочую область для sys1T, то увидим, что матрицы для этой системы такие же, как для sys1, но имеется дополнительное указание о наличии запаздывания на входе и выходе. Для MIMO системы воз-

можно задать различные запаздывания для каждого канала входа и выхода. Подробно эти и другие команды рассматриваются в последующих главах.

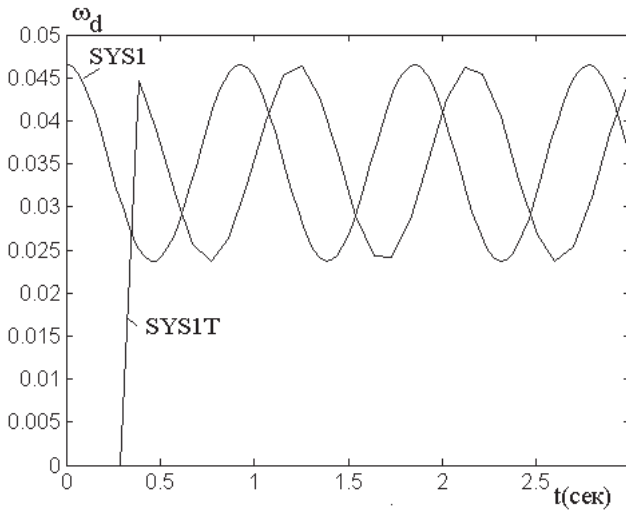


Рис. 1.3. Импульсная реакция в системе без и с запаздыванием

Control System Toolbox работает с дискретными системами практически так же, как и с непрерывными, только при их формировании дополнительно указывается время выборки (период квантования) T_s. В результате образуется система, описываемая рекуррентными соотношениями

$$\mathbf{x}(k+1) = \mathbf{A}_d\mathbf{x}(k) + \mathbf{B}_d\mathbf{u}(k) \qquad \mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k), \tag{1.23}$$

где все величины фиксируются в моменты kT_s, $k = 0,1,...$ Непрерывную модель можно преобразовать в дискретную, используя команду **c2d(sys,Ts).** При этом предполагается, что на входе непрерывной системы устанавливаются фиксаторы нулевого порядка, а выходы фиксируются в дискретные моменты времени. Дополним файл (П. 1.1) командами

```
Ts = 0.1;
sys1d = c2d(sys1, Ts)
pad = eig(sys1d)
impulse(sys1d, 'k').
```
(П. 1.1.7)

В результате в командном окне и в рабочей области зафиксирована дискретная система с ее матрицами, в переменной pad находятся полюса системы 1.0, 0.7786 + 0.6275i, 0.7786 – 0.6275i, указывающие на неустойчивость (точнее, условную устойчивость) системы, так как модуль комплексных корней практически равен 1, а на графике рис. 1.4 показан процесс при импульсном воздействии. Существуют и другие методы преобразования непрерывной системы в дискретную, которые будут рассмотрены далее.

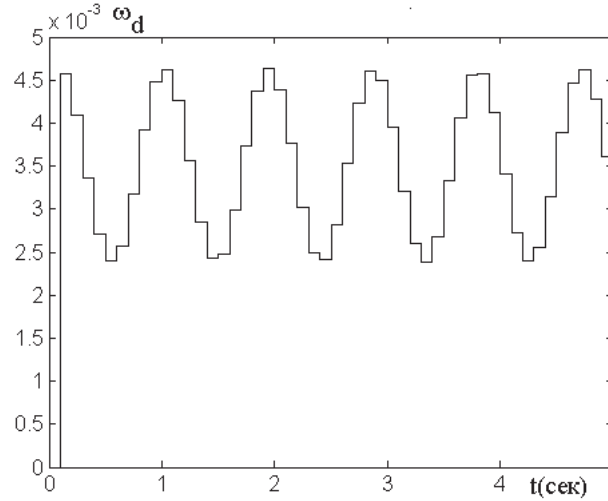


Рис. 1.4. Процессы в дискретной системе

Обратное преобразование дискретной системы в непрерывную осуществляется командой **d2c.** Применив эту команду к sys1d:

sys11= d2c(sys1d), (П. 1.1.8)

получим систему, которая отличается от sys1 тем, что матрицы **a** и **b** вместо нулевых элементов содержат весьма малые числа порядка 10^{-15} и менее.

В Robust Control Toolbox для преобразования непрерывной системы в дискретную и обратно используется билинейное преобразование, имеющее вид:

$$s = (\alpha z + \delta)/(\gamma z + \beta), \qquad (1.24)$$

где s — аргумент преобразования Лапласа, z — аргумент дискретного преобразования Лапласа, α, δ, γ, β — коэффициенты, опреде-

ляющие вид билинейного преобразования. Всего предусмотрено 7 видов этого преобразования, но мы будем рассматривать только два: преобразование 'TUSTIN' и 'BwdRec' — обратный метод прямоугольников. В первом случае

$$s = \frac{2}{T_s}\frac{z-1}{z+1}, \qquad z = \frac{1+0.5T_s s}{1-0.5T_s s}, \tag{1.25}$$

а во втором

$$s = \frac{z-1}{T_s z}, \qquad z = \frac{1}{1-T_s s} \tag{1.26}$$

где T_s — период выборки. Соответствующая команда записывается как

[ssd] = bilin(sys,ver,'Type',aug),

где sys — исходная система (непрерывная или дискретная), ssd — преобразованная система (дискретная или непрерывная), *ver* равно 1 при преобразовании непрерывной системы в дискретную и равно –1 при обратном преобразовании, *aug* — время выборки в секундах, 'Type' — строка с именем типа преобразования.

Выполним в (П. 1.1) команды

```
sys2d = bilin(sys1, 1, 'TUSTIN', 0.1);
sys3d = bilin(sys1, 1, 'BwdRec', 0.1);
pad2 = eig(sys2d)
pad3 = eig(sys3d).
```
(П. 1.1.9)

Получим следующие значения полюсов дискретных систем: для sys2d 1.0; 0.7937 ± 0.6084i; для sys3d 1.0; 0.6849 ± 0.4646i. Видно, что полюса sys2d достаточно близки к полюсам sys1d, найденным ранее, тогда как полюса sys3d отличаются существенно. Таким образом, преобразование 'TUSTIN' оказывается более точным.

Теперь преобразуем дискретную систему sys1d, полученную выше, в непрерывную, используя команду

sys1c = bilin(sys1d, -1, 'TUSTIN', 0.1). (П. 1.1.10)

Матрица **A** этой системы оказывается равной

```
A = [-6.587e -016   -0.04838   -1.11e -016; 252.8  6.217e -015  -252.8;...
     -2.289e -017     0.1486   -4.441e -015]
```

и отличается незначительно от матрицы **a** исходной непрерывной системы.

1.2. Использование передаточных матриц

Перейдем к описанию динамических систем в виде передаточной матрицы (для SISO — передаточной функции). Для MIMO системы *ij*-тый элемент передаточной матрицы — передаточная функция от *j*-того входа к *i*-тому выходу. Передаточная матрица получается на основании (1.1), (1.2) как

$$\mathbf{F}(s) = \mathbf{D} + \mathbf{C}\,(s\mathbf{I} - \mathbf{A})^{-1}\,\mathbf{B}. \qquad (1.27)$$

Дополним файл (П. 1.1) командами

```
systr=tf(sys1)
systrd=tf(sys1d),
```
(П. 1.1.11)

выполним его и получим выражения для передаточных функций $\omega_d(s)/M_d(s)$ для непрерывного и дискретного вариантов нашей системы:

$$(0.04651\ s^2 + 1.615)/(s^3 + 46.02\ s), \qquad (1.28)$$

$$(0.004566z^2 - 0.007577z + 0.004566)/(z^3 - 2.557z^2 + 2.557z - 1). \qquad (1.29)$$

Для работы с системой в таком виде создадим новый файл (П. 1.2). Часто бывает необходимо с целью дальнейшего анализа по уравнениям состояний системы, коэффициенты которых являются буквенными обозначениями параметров, получить выражение для передаточных функций тоже в зависимости от параметров. Для этой цели можно воспользоваться элементами символьной математики MATLAB. Для нашего примера напишем программу (для других систем уравнений буквенные обозначения параметров, естественно, будут другими):

```
syms  s J1 J2 C ag bg F R
ag = [0  -1/J1  0; C  0  -C; 0  1/J2  0];
bg = [1/J1; 0; 0]; cg = [1  0  0];dg = 0;
R = inv(s*eye(3) - ag);
F = cg*R*bg.
```
(П. 1.2)

В результате ее выполнения получим:

$$F = (s^2J_2 + C)/[s(J_1J_2s^2 + J_1C + J_2C)]. \qquad (1.30)$$

Дополним (П. 1.2):

```
num = [J2  0  C]
den = [J1*J2  0  C*(J1+J2) 0]
J1 = 21.5;J2 = 7;C =243;
num = subs(num);
```

```
den = subs(den);
sys1Tr = tf(num,den)
sys1Tr1 = tf([J2 0 C],[J1*J2 0 C*(J1+J2) 0]).
```
(П. 1.2.1)

На основании (1.30) составляются векторы коэффициентов числителя num и знаменателя den, причем коэффициенты располагаются по нисходящим степеням *s*, затем с помощью команды **subs** вычисляются значения элементов векторов, и команда **sys1Tr = tf(num,den)** формирует систему с приведенной выше передаточной функцией *F*. Можно также векторы числителя и знаменателя указывать прямо в команде, как это сделано для sys1Tr1.

Если использование буквенных обозначений не предполагается, то для преобразования системы *SS* в систему *TF* можно воспользоваться командой **ss2tf(sys)**. Тогда вместо (П. 1.2.1) можно записать:

```
J1 = 21.5;J2 = 7;C = 243;
ag = [0  -1/J1  0; C 0 -C;0  1/J2  0];
bg = [1/J1; 0; 0]; cg = [1 0 0];
[num, den] = ss2tf(ag, bg, cg, 0)
sys1Tr = tf(num, den) .
```
(П. 1.2.2)

Полученная передаточная функция **sys1Tr** отличается от найденной в (П. 1.2.1) только тем, что у нее старший коэффициент знаменателя равен 1 как в (1.28).

Имеется возможность также непосредственно приводить в программе выражение передаточной функции системы как функции *s*, если предварительно определить:

```
s = tf('s').
```

Пример использования этой возможности будет приведен в этом разделе далее. Обратим также внимание на эффективность использования операции свертки *conv*. Если, например, числитель состоит из произведения двух полиномов с коэффициентами [1 1] и [3 2 1] соответственно, то вместо вычисления произведения этих двух полиномов можно записать

```
[num] = conv([1 1], [3 2 1]).
```

Для MIMO системы *ij*-тый элемент передаточной матрицы F(*s*) задается таким же образом, а затем формируется матрица передаточных функций

$$\mathbf{F}(s) = [F_{11}(s)\ F_{12}(s)...;F_{21}(s)\ F_{22}(s)...;...]. \qquad (1.31)$$

Если **F**(*s*) сформирована, то можно работать с отдельными ее элементами. Например, команда **F(1, 2)** извлекает передаточную функцию между первым выходом и вторым входом.

Возможно также образовать массив систем, с которым можно обращаться как с одним объектом. Пусть, например, нужно исследовать влияние какого-либо параметра на переходный процесс. Вычислим значение матрицы **A** при различных значениях этого параметра $\mathbf{A}_1$,$\mathbf{A}_2$, ..., образуем массив с элементами

```
sys(:,:,1) = ss(A1, B, C, D);
sys(:,:,2) = ss(A2, B, C, D);
```

и добавим команду

```
step(sys).
```

В результате на одном и том же графике будет построено семейство переходных процессов для всех систем, входящих в массив.

Со сформированной командой **tf** системой можно действовать таким же образом, как с системой, сформированной командой **ss,** при наличии различий они будут упомянуты при описании конкретных команд. Например, sys1Tr можно преобразовать в дискретную систему командой **c2d**:

```
sys1Trd = c2d(sys1Tr, 0.1),
```
(П. 1.2.3)

получим систему с дискретной передаточной функцией, уже приведенной выше (1.29).

В некоторых случаях, например, при использовании полученных результатов для программирования цифровых сигнальных процессоров более удобно иметь дискретную передаточную функцию по нисходящим обратным степеням z. Для этой цели используется команда

```
filt(numz,denz).
```
(П. 1.2.4)

Если, например, использовать numz,denz из (1.29), то после выполнения этой команды получим:

$$(0.004566\ z^{-1} - 0.007577\ z^{-2} + 0.004566\ z^{-3})/$$

$$/(1 - 2.557\ z^{-1} + 2.557\ z^{-2} - z^{-3}). \qquad (1.32)$$

Определенную специфику имеет задание запаздывания для систем в виде передаточной функции. В этом случае система не различает входные и выходные задержки, есть только общее запаздывание:

$$Ft(s) = F(s)e^{-pTs}. \qquad (1.33)$$

Значение запаздывания задается либо отдельной командой для уже созданной системы

```
sys1Trt = set(sys1Tr, 'ioDelay', 0.1),
```
(П. 1.2.5)

либо таким же текстом при создании системы, значение запаздывания задается в секундах. Дополним файл (П. 1.2) этой командой и выполним его. Получим в рабочей области

$$e^{-0.1s}\,\frac{7s^2 + 243}{150.5s^3 + 6926s}. \qquad (1.34)$$

Для дискретной системы запаздывание кратно периоду квантования. Ведем в файл (П. 1.2) команду

```
sys1Trdt = set(sys1Trd, 'ioDelay', 3)
```
(П. 1.2.6)

и выполним его. Получим в рабочей области

$$F_{dt}(z) = z^{-3}\,\frac{0.004566z^2 - 0.007577z + 0.004566}{z^3 - 2.577z^2 + 2.557z - 1}. \qquad (1.35)$$

С передаточной функцией непрерывной системы, содержащей экспоненту, не всегда удобно иметь дело, например, в замкнутых системах при расчете устойчивости, поэтому в Control System Toolbox предусмотрена возможность замены экспоненты аппроксимацией Паде, под которой понимают отношение двух полиномов, имеющее амплитудно-частотную характеристику, равную 1, и фазовую характеристику, близкую к характеристике звена чистого запаздывания. Степень полиномов N определяет порядок аппроксимации. Например, при N = 2

$$e^{-sT_s} \approx \frac{1 - 0.5T_s + 0.083T_s^2}{1 + 0.5T_s + 0.083T_s^2}. \qquad (1.36)$$

Чем выше порядок аппроксимации, тем выше точность и тем больше значение допустимого при такой аппроксимации запаздывания, но тем сложнее получающиеся при этом передаточные функции.

На рис. 1.5 приведены зависимости ошибки в воспроизведении фазо-частотной характеристики звена чистого запаздывания функцией Паде в зависимости от порядка функции. Если, например, считать допустимой ошибкой 0.1 рад, то для N = 1 произведение ωT_s в рабочем диапазоне не должно превосходить 1.12, для N = 2—2.56, для N = 3—4.15, для N = 4—5.82. Ниже приводится программа, по которой рассчитаны эти зависимости:

```
clear
for N = 1:4;
g = 15*N;
[nump, denp] = pade(1, N)
for j = 1:g
```

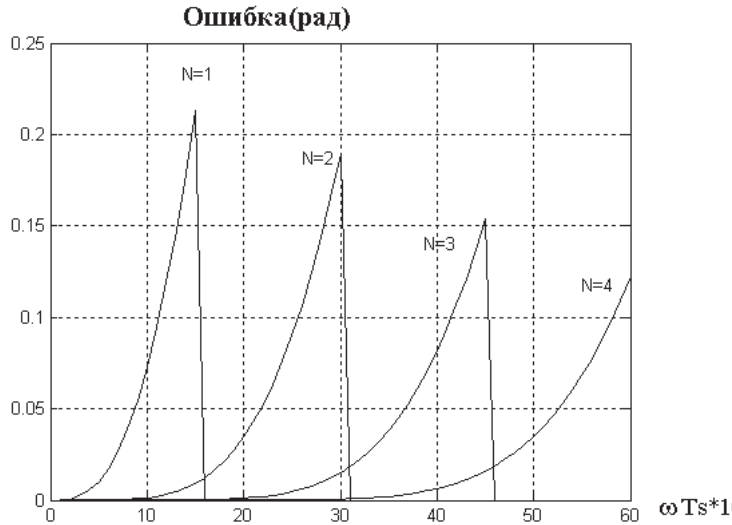


Рис. 1.5. Фазовая ошибка аппроксимации чистого запаздывания

```
c(j) = j*0.1;
s(j) = i*c(j);
na(j,N) = polyval(nump,s(j));
da(j,N) = polyval(denp,s(j));
delta(j,N) = angle(na(j,N)/da(j,N));
if delta(j,N) < 0
dd(j,N) = delta(j,N) + c(j);
else
dd(j,N) = delta(j,N) + c(j) - 2*pi;
end;end;end;
plot(dd)
grid
gtext('N = 1')
gtext('N = 2')
gtext('N = 3')
gtext('N = 4').
```

(П. 1.3)

Введем в файл (П. 1.2) команду

```
sysx = pade(sys1Tr,2)
```

(П. 1.2.7)

и выполним его (файл). Получим в рабочем пространстве вместо (1.34)

$$F(s) = \frac{7s^4 - 420s^3 + 8643s^2 - 14580s + 291600}{150.5s^5 + 9030s^4 + 187555s^3 + 415530s^2 + 8311000s}. \quad (1.37)$$

Оглавление