РЕШЕНИЕ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ С ИСПОЛЬЗОВАНИЕМ MATLAB

Л. Ф. Шампайн, И. Гладвел, С. Томпсон

ЛАНЬ®

Л. Ф. ШАМПАЙН, И. ГЛАДВЕЛ,
С. ТОМПСОН

РЕШЕНИЕ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ С ИСПОЛЬЗОВАНИЕМ MATLAB

УЧЕБНОЕ ПОСОБИЕ

Перевод с английского
И. А. МАКАРОВА

САНКТ-ПЕТЕРБУРГ · МОСКВА · КРАСНОДАР ·
2009

ББК 22.161.6я73
Ш 19

Шампайн Л. Ф., Гладвел И., Томпсон С.

Ш 19 Решение обыкновенных дифференциальных уравнений с использованием MATLAB: Учебное пособие / Пер. с англ. И. А. Макарова. — СПб.: Издательство «Лань», 2009. — 304 с.: ил. — (Учебники для вузов. Специальная литература).

ISBN 978-5-8114-1033-0

В учебном пособии представлены все разделы информатики, определяющие современный уровень подготовки.

В книге исследуются вопросы численного решения дифференциальных уравнений с использованием системы MATLAB. Рассматриваются задачи с начальными условиями (ЗНУ) и граничными условиями (ЗГУ) для обыкновенных дифференциальных уравнений, а также дифференциальные уравнения с запаздывающим аргументом (ДУЗА). Каждой из этих трех тем посвящена отдельная глава, имеющая следующую структуру. В начале каждой главы представлены теоретические результаты, лежащие в основе решения рассматриваемого класса задач для дифференциальных уравнений. После иллюстрации изложенного в начале главы теоретического материала физически мотивированными примерами, разрабатываются соответствующие численные методы, при рассмотрении которых основное внимание уделяется только тем теоретическим аспектам, которые имеют важное значение при практическом применении и программной реализации данного метода. В заключение каждой из глав приведены практические руководства, основу которых составляют решения различных математических, физических, биологических и других задач.

Авторы книги без излишнего углубления в теоретические основы современных численных методов решения дифференциальных уравнений знакомят читателя с особенностями использования алгоритмических реализаций этих методов, что должно способствовать принятию правильного решения в сложных ситуациях, возникающих на практике при компьютерном исследовании поведения численных решений различных дифференциальных уравнений. Книга будет полезна студентам высших учебных заведений, специализирующихся по техническим и физико-математическим специальностям, а также исследователям в области математического моделирования физических, химических, биологических и экономических систем.

ББК 22.161.6я73

Обложка
А. Ю. ЛАПШИН

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ ПЕРЕВОДЧИКА

Многие физические законы, которым подчиняются те или иные явления, могут быть записаны в виде дифференциальных уравнений. Эти уравнения описывают изменение соответствующих физических величин с течением времени и могут служить в качестве математической модели соответствующего процесса. Дифференциальные уравнения играют важную роль в прикладной математике, физике и в других науках, таких как биология, экономика и электротехника; в действительности, они возникают везде, где есть необходимость количественного (числового) описания явлений окружающего мира.

Теория численного решения дифференциальных уравнений хорошо разработана и на ее основе создано множество прикладных программ, позволяющих пользователю получить решение и вывести его в графическом виде. Среди этих программ следует в первую очередь отметить такие математические пакеты, как MATLAB, MATHEMATICA, MAPLE и MATHCAD. Существует обширная библиография по численным методам и множество руководств пользователя по использованию указанных программных средств. Предлагаемая вниманию читателя книга представляет собой уникальное издание, наводящее мосты между чистой теорией и практикой ее применения.

Указанные выше компьютерные программы представляют собой мощные средства численного анализа, снабженные удобным интерфейсом. У неопытного пользователя может возникнуть иллюзия, что эти инструменты гарантированно обеспечивают получение достаточно точного и достоверного решения любого рассматриваемого дифференциального уравнения при любом выборе используемого численного метода и произвольно заданных параметрах вычислительного процесса. Однако, как показывает практика, при численном решении сложных дифференциальных уравнений могут возникать ситуации, когда для получения правильной аппроксимации точного решения одновременно необходимо глубокое понимание существа используемого численного метода, теоретических основ теории дифференциальных уравнений и физики исследуемой задачи. Представленная книга позволяет читателю выработать не только необходимые практические навыки в сведении воедино указанных трех аспектов численного исследования дифференциальных уравнений, но и специфическое мышление, позволяющее эффективно выбрать правильный путь решения задачи. Авторы обладают неоценимым опытом в этой области и на страницах своей книги щедро делятся им со своими читателями.

И. А. Макаров
канд. физ.-мат. наук

ПРЕДИСЛОВИЕ

Эта книга предназначена для студентов, преподавателей, инженеров и научных работников, чья учебная и исследовательская деятельность связана с решением обыкновенных дифференциальных уравнений (ОДУ). В ней рассматриваются методы решения ОДУ с начальными условиями (ЗНУ), ОДУ с граничными условиями (ЗГУ), а также методы решения дифференциальных уравнений с запаздывающим аргументом (ДУЗА). Книга «Решение ОДУ с использованием MATLAB» представляет собой учебный курс длительностью один семестр для студентов инженерных и физико-математических специальностей. Для понимания материала книги учащиеся должны обладать базовыми знаниями в области теории ОДУ, теории матриц и численных методов. Подразумевается, что студенты обладают определенными навыками программирования, предпочтительно в среде MATLAB. Эта книга может также служить в качестве справочного руководства для инженеров и научных специалистов. Читатели могут ознакомиться с существом рассматриваемых в книге задач, используя детально рассмотренные примеры и их решения. Представленные в этой книге программы могут использоваться читателями при написании собственных программ.

Три темы, рассматриваемые в этой книге, обычно изучаются в рамках трех отдельных курсов длительностью в один семестр. Книга «Решение ОДУ с использованием MATLAB» объединяет под одной обложкой все эти три тематики и изложение их основ, необходимых для решения рассматриваемых задач, занимает около 300 страниц. Это оказалось возможным благодаря соответствующему выбору глубины детализации представленного теоретического материала, а также вследствие концентрации внимания читателя на решении конкретных задач. Книга начинается с главы, имеющей вводный характер. Вторая глава посвящена решению задач с начальными условиями (ЗНУ). Читатель должен ознакомиться с материалом этих двух глав в порядке их изложения, но оставшиеся две главы, посвященные решению задач с граничными условиями ЗГУ и решению дифференциальных уравнений с запаздывающим аргументом (ДУЗА), могут изучаться независимо друг от друга. Материал каждой из этих двух последних глав вполне может быть изложен в течение одного семестра, однако весь материал книги можно представить за тот же срок лишь при соответствующей подготовке студентов. Материал главы, посвященной ДУЗА, может быть изложен достаточно быстро, поскольку в ней рассматривается лишь один из подходов к решению подобных задач, являющийся обобщением методов, рассмотренных в главе, посвященной ЗНУ. Каждая глава начинается с обсуждения практических примеров соответствующей задачи из реальной жизни. После этого приводятся наиболее часто используемые численные

методы решения этой задачи. Обсуждение каждого метода представлено в краткой форме и детальное изложение теоретических основ опускается, однако некоторые аспекты, являющиеся важными для практической реализации соответствующего метода, рассматриваются более подробно. Очень часто оказывается, что для решения рассматриваемой задачи необходимо сделать нечто большее, нежели просто написать программный код с вызовом соответствующей процедуры. Этот аспект решения задач рассматривается в заключительном параграфе каждой из глав, в котором представлено практическое руководство по решению практических задач.

Существует несколько сред программирования, предназначенных для решения обыкновенных дифференциальных уравнений. В нашей книге тексты программ в примерах и упражнениях приведены для системы MATLAB. Использование этой системы обусловлено тем, что в последнее время она стала широко использоваться в учебных заведениях и при научных исследованиях. Вычислительные процедуры в MATLAB являются очень эффективными. Более того, они имеют единый «дизайн», что облегчает их использование и способствует быстрому обучению работе в среде MATLAB. Язык программирования MATLAB является языком высокого уровня, что позволяет писать короткий программный код, примеры которого могут быть полностью приведены в тексте книги. Поскольку тексты программ представлены в электронном виде, они могут служить в качестве шаблонов при написании других программ, предназначенных для решения аналогичных задач. В частности, в некоторых из представленных в этой книге упражнений студентам предлагается модифицировать программы, рассмотренные в основном тексте книги в качестве примеров. Тексты этих программ доступны на сайте для свободного скачивания: http://www.radford.edu/~thompson/sodeswm/SolvingODESwithMatlab.html. Построение графиков является неотъемлемой частью среды программирования MATLAB, и поэтому исследование полученного решения обычно выполняется с их использованием. MATLAB позволяет выполнять символьные вычисления, благодаря наличию в этой системе ядра другой среды программирования — Maple. В книге «Решение ОДУ с использованием MATLAB» эти возможности используются при анализе и решении некоторых примеров и упражнений.

Первая процедура решения ОДУ, разработанная для системы MATLAB, была основана на программном коде, который был написан на языке FORTRAN Ларри Шампайном (L.F.Shampine) и Г.А. Ваттом (H.A.Watt). Клив Моулер (Cleve Moler) стал инициатором долгого и продуктивного сотрудничества Шампайна с компанией The MathWorks — разработчиком среды программирования MATLAB. В результате проведенной исследовательской работы [Shampine & Mark Reichelt, 1997] был создан специализированный набор инструментальных средств MATLAB — ODE Suite. ODE Suite получил дальнейшее развитие благодаря работе [Shampine, Reichelt & Jacek Kierzenka, 1999], а также в результате эволюции самой системы MATLAB. В частности, некоторые процедуры решения ЗНУ стали применимы при решении дифференциально-алгебраических уравнений (ДАУ) индекса 1, возникающих при решении систем, матрица весовых коэффициентов которых сингулярна. Впоследствии Кирженка и Шампайн (2001) добавили в этот набор инструментальных средств процедуру решения ЗГУ. В последнее время Скип Томпсон (Skip Thompson), Шампайн и Кирженка расширили MATLAB ODE Suite процедурой решения ДУЗА с постоянными запаздываниями [Shampine & Thompson, 2001].

Мы выражаем глубокую благодарность Кливу, Марку и Яцеку за представившуюся в свое время возможность работать с ними над программным обеспечением для ранних версий системы MATLAB, которое впоследствии легло в основу программ, рассмотренных в книге «Решение ОДУ с использованием MATLAB».

Каждый из авторов имеет многолетний опыт решения ОДУ в промышленных, научных и учебных организациях. За время работы нами было опубликовано более 200 статей и несколько книг, посвященных этим вопросам, но мы всегда хотели написать книгу, которая была бы доступна для понимания широкой аудитории и в которой был бы суммирован весь наш практический опыт. Книга «Решение ОДУ с использованием MATLAB» стала воплощением этой мечты. Мы благодарны за помощь, которую нам оказали множество экспертов, высказавших полезные комментарии по поводу некоторых результатов, приведенных в нашей книге. В этой связи хотелось бы упомянуть Вайна Енрайта (Wayne Enright) и Яцека Кирженка, чьи суждения были особенно полезны.

Л.Ф. Шампайн, И. Гладвел, С. Томпсон

Глава 1

ВВОДНАЯ ГЛАВА

§ 1.1. ВВЕДЕНИЕ

Обыкновенные дифференциальные уравнения (ОДУ) используются при исследовании инженерных и научных задач в качестве модели, описывающей процесс изменения физических величин. Поэтому вводный курс по решению обыкновенных дифференциальных уравнений является обязательной частью учебных планов для соответствующих специальностей. Подобные учебные курсы обычно дают общее представление о существе предмета, однако рассматриваемые в них методы часто оказываются неприменимы в практических задачах, в которых требуется найти решение больших, сложных и нелинейных систем уравнений. Эта книга посвящена задаче численного решения ОДУ. Ее авторы обладают многолетним опытом оказания консультационных услуг в промышленных, научных и учебных организациях. Эта глава начинается с обсуждения задачи численного решения ОДУ с использованием стандартных численных методов, а также программных средств, позволяющих получить это решение на компьютере. В последующих главах мы обсудим наиболее популярные численные методы решения для основных классов ОДУ. Представленные в книге многочисленные примеры, иллюстрирующие теоретический материал, помогают понять, как решаются задачи на практике. Система MATLAB является очень удобной и широко применяемой *средой программирования* (СП), в которой почти все эти задачи могут быть решены с использованием очень эффективных численных процедур. В этой системе применяется язык программирования высокого уровня, что позволяет писать программы с очень коротким кодом, который может быть полностью приведен в книгах и статьях. Каждый из рассмотренных нами примеров сопровождается листингом соответствующей программы. В нашей книге также кратко упоминаются некоторые другие среды программирования, применимые для решения ОДУ. Каждый из авторов этой книги являлся автором программ решения ОДУ, нашедших широкое применение при выполнении прикладных и научных расчетов.

Обыкновенные дифференциальные уравнения представляют собой математические отношения между функциями и их производными. Одним из таких уравнений является линейное ОДУ

$$y'(t) = y(t), \tag{1.1}$$

которое задано, скажем, на промежутке времени $0 \leqslant t \leqslant 10$. Из начального курса математического анализа известно, что для нахождения решения дифференциального уравнения недостаточно знать лишь само это уравнение. Для определения

решения необходимо также задать и его начальное значение. Например, существует единственное решение ОДУ (1.1), для которого $y(0) = 1$, и это решение имеет вид $y(t) = e^t$. Рассмотренная выше задача называется задачей с начальными условиями (ЗНУ) для ОДУ. Так же как и в этом примере, возникающие на практике ЗНУ имеют одно и только одно решение. Иногда решения определяются более сложным образом. Подобные ситуации очень важны в приложениях, однако они часто не рассматриваются в рамках начальных курсов по математическому анализу (возможно, исключая задачу Штурма–Лиувилля). Предположим, что $y(x)$ удовлетворяет уравнению

$$y''(x) + y(x) = 0, \tag{1.2}$$

при $0 \leqslant x \leqslant b$. В случае, когда решение этого ОДУ определяется условиями на обоих концах интервала

$$y(0) = 0, \; y(b) = 0,$$

соответствующая задача будет называться краевой задачей, или задачей с граничными условиями (ЗГУ). Принадлежащая к этому классу задача Штурма–Лиувилля всегда имеет тривиальное решение $y(x) \equiv 0$, но при некоторых значениях b существуют и нетривиальные решения. Например, при $b = 2\pi$ рассматриваемая ЗГУ имеет бесконечно много решений вида $y(x) = \alpha \sin(x)$, где α — любая константа. В отличие от ЗНУ, обычно имеющих единственное решение, возникающие в практических приложениях ЗГУ могут не иметь решения, иметь единственное решение, или иметь более одного решения. В последнем случае, количество имеющихся решений может быть конечным или бесконечным.

Уравнение (1.1) показывает, что скорость изменения решения в момент времени t равна значению этого решения в этот момент времени. Во многих физических системах изменение с течением времени значения решения происходит с некоторой задержкой и математические модели подобных систем описываются дифференциальными уравнениями с запаздывающим аргументом (ДУЗА). Часто в качестве такого запаздывания выступает постоянная величина. Например, если система описывается ОДУ (1.1), то ее модель с запаздыванием, равным одной единице времени, может быть записана в виде ДУЗА

$$y'(t) = y(t-1), \tag{1.3}$$

заданного при $0 \leqslant t \leqslant 10$. Таким образом, рассматриваемый случай сводится к стандартному ОДУ, т. е. при постоянной величине запаздывания теория ДУЗА и методы численного решения подобных задач могут быть основаны на соответствующих результатах для ОДУ. Тем не менее, следует отметить существенные отличия. В случае ОДУ (1.1) для определения решения достаточно задать его начальное значение $y(0) = 1$, однако этого недостаточно при нахождении решения ДУЗА (1.3). Действительно, при $t = 0$ для определения $y'(0)$ нам необходимо знать значение $y(-1)$, однако эта величина представляет собой значение решения в момент, предшествующий начальному моменту времени. Таким образом, для решения задачи с начальными условиями для ДУЗА (1.3) необходимо знать не только значение решения в начальный момент времени, но и его предысторию, задаваемую

функцией истории. В рассматриваемом примере достаточно задать функцию истории $y(t)$ при $-1 \leqslant t \leqslant 0$ и тогда задача с начальными условиями будет иметь единственное решение.

В этой книге рассматриваются методы решения для задач с начальными и граничными условиями для ОДУ, а также для задач с начальными условиями для класса ДУЗА с постоянным запаздыванием. Для краткости изложения мы будем использовать соответствующие аббревиатуры: ЗНУ, ЗГУ и ДУЗА. Далее в этой главе мы обсудим основополагающие вопросы, которые являются общими для этих трех задач. Некоторые из этих вопросов действительно являются настолько фундаментальными, что глубокое понимание их существа оказывается необходимым даже при решении лишь конкретного уравнения. Соответствующие теоретические материалы для ЗНУ рассматриваются в главе 2, для ЗГУ — в главе 3, для ДУЗА — в главе 4. Глава, посвященная ЗНУ, предшествует остальным главам, поскольку в ней излагаются основные идеи и рассматриваются программные средства, применяющиеся на протяжении всей книги. Главы, посвященные ЗГУ и ДУЗА, являются независимыми.

Предполагается, что читатель знаком с элементами программирования в среде MATLAB, и поэтому мы будем рассматривать только те вопросы, которые имеют непосредственное отношение к решению ОДУ. Если вам необходимо расширить свои знания в области используемого в MATLAB языка программирования, вы может воспользоваться документацией, прилагаемой к этой СП, или обратиться к многочисленным книгам, в которых описан MATLAB. В качестве одной из подобных книг мы рекомендуем «Руководство пользователя MATLAB» (MATLAB Guide, Higham&Higham, 2000). Большинство программ, представленных в нашей книге, позволяют вывести цветные графики решений. Поскольку использование цветных иллюстраций в книгах нецелесообразно, мы модифицировали эти программы так, чтобы получаемые графики решений были монохромными. Для запуска программ из главы 4 необходимо использовать MATLAB версии 6.5, релиз 13 (или более поздние версии), для запуска программ из остальных глав можно использовать MATLAB версии 6.1. Большинство программ для научных вычислений, на которые мы ссылаемся в нашей книге, основаны на коде из следующих библиотек программ: NAG (2002), Visual Numerics (IMSL 2002), Harwell 2000 (*H2KL*) и Netlib Repository (*Netlib*). Если первоисточник программного кода явно не указан, он может быть определен с использованием системы классификации GAMS (Guide to Available Mathematical Software — Руководство по существующему математическому программному обеспечению).

При исследовании и анализе решений математических задач численные методы и аналитические средства классической прикладной математики взаимно дополняют друг друга. Решение дифференциальных уравнений в аналитическом виде возможно найти лишь в простейших случаях. Для решения подобных задач можно воспользоваться программными средствами компьютерной алгебры такими, как Maple (1998) или Mathematica (Wolfram, 1996). Аналитические решения, приведенные в примерах из нашей книги, были получены с использованием модуля символьных вычислений Maple, имеющегося в студенческой версии MATLAB, или с использованием набора программных средств для символьных вычислений (MATLAB Symbolic Toolbox), имеющегося в профессиональной версии MATLAB.

Следует отметить, что даже незначительное изменение уравнения может существенно усложнить задачу нахождения решения в аналитическом виде. Например, для получения решения ОДУ

$$y' = y^2$$

с использованием модуля символьных вычислений Maple в среде MATLAB необходимо в командной строке набрать

```
>> y = dsolve('Dy = y^2')
y = -1/(t-C1)
```

(Иногда мы немного редактируем листинги для получения более короткого их представления в тексте книги.) В полученном выражении для общего решения величина `C1` является произвольной константой. Это семейство решений, выраженное в терминах известных функций, позволяет нам предсказать поведение конкретных решений. Если «немного» изменить рассматриваемое ОДУ

$$y' = y^2 + 1,$$

то общее решение, найденное с использованием функции **dsolve**, может быть записано в виде

```
y = tan(t+C1)
```

Легко видеть, что полученное выражение для решения является более сложным, чем в предыдущем случае, но по-прежнему представляется в терминах хорошо известных функций. Решение еще более усложненной версии рассматриваемого ОДУ

$$y' = y^2 + t,$$

найденное с использованием той же функции **dsolve**, имеет вид

```
y = (C1*AiryAi(1,-t)+AiryBi(1,-t))/
    (C1*AiryAi(-t)+AiryBi(-t))
```

Это выражение может быть представлено в стандартной математической записи

$$y(t) = \frac{C_1 Ai'(-t) + Bi'(-t)}{C_1 Ai(-t) + Bi(-t)},$$

где $Ai(t)$ и $Bi(t)$ — функции Эйри. (В модуле символьных вычислений Maple для обозначения этих функций используются имена **AiryAi** и **AiryBi**. Справку по их использованию можно получить, набрав, например, команду **mhelp airy**. В MATLAB используются другие имена этих функций и соответствующую справку можно получить, набрав, например, команду **help airy**.) Как и ранее, `C1` — произвольная константа. Функции Эйри не столь широко известны как функция тангенса. Полученные выше решения полезны при исследовании поведения решений аналитическими методами, однако для получения представления о поведении этих решений часто оказывается достаточным нарисовать их графики при

различных начальных значениях. Изменив в очередной раз рассматриваемое ОДУ

$$y' = y^2 + t^2,$$

получаем с использованием dsolve общее решение вида

```
y = -t*(C1*besselj(-3/4,1/2*t^2)+bessely(-3/4,1/2*t^2))/
       (C1*besselj(1/4,1/2*t^2)+bessely(1/4,1/2*t^2))
```

В стандартной математической записи оно имеет следующий вид

$$y(t) = -t\frac{C_1 J_{-3/4}\left(\frac{t^2}{2}\right) + Y_{-3/4}\left(\frac{t^2}{2}\right)}{C_1 J_{1/4}\left(\frac{t^2}{2}\right) + Y_{1/4}\left(\frac{t^2}{2}\right)}.$$

Легко видеть, что также как и в предыдущем случае, решение выражается в терминах специальных функций — функций Бесселя дробного порядка. Для получения представления о поведении этих решений необходимо нарисовать их графики. Эти решения будут рассмотрены позднее в примере 2.3.1.

Если изменить в рассматриваемом ОДУ степень y, мы получим совершенно другой результат[1]

```
>> y = dsolve('Dy = y^3 + t^2')
Warning: Explicit solution could not be found.
```

Этот пример показывает, что даже простые на первый взгляд уравнения могут не иметь решения, представимого в терминах функций, реализованных в модуле компьютерной алгебры Maple. Подобные ситуации возникают нередко и обычно Maple не может определить решение в аналитической форме в ситуациях, когда это решение действительно не может быть найдено. На практике явное выражение для решения системы ОДУ может быть получено лишь в некоторых специальных случаях.

Для рассмотренных выше достаточно простых скалярных ОДУ можно использовать модуль символьных вычислений и получить соответствующие решения в аналитической форме. Далее мы обсудим отличия между аналитическими и численными решениями ОДУ. Аналитические решения, представленные в рассмотренных примерах, позволяют получить общее представление об общем решении, однако для лучшего понимания эволюции решений исследуемого ОДУ полезно нарисовать графики некоторых его частных решений. Для получения значений специальных функций, присутствующих в записи аналитического решения, необходимо применить существующие вычислительные процедуры. Однако если в подобных ситуациях приходится использовать численные методы, почему бы нам не отказаться от предварительного получения решения в аналитической форме и всегда находить его численно? Непосредственное использование численных методов для

[1] Прим. перев. — Перевод сообщения в нижеприведенном листинге гласит: «Предупреждение: решение в явном виде не может быть найдено.»

получения решений может быть приемлемым подходом при исследовании частных случаев ЗНУ, однако в общем случае применение численных методов имеет некоторые ограничения. Действительно, численное вычисление значений специальных функций (например, функций Эйри и Бесселя) может вызвать определенные трудности, т. к. эти функции могут иметь сингулярности или осциллировать с высокой частотой. Если решение исследуемого ОДУ характеризуется подобным поведением, или если мы заинтересованы в исследовании поведения этого решения при $t \to \infty$, мы не сможем воспользоваться численными методами для непосредственного нахождения этого численного решения. Получение решения в аналитической форме позволяет изолировать указанные проблемы, в результате чего вычисление точного решения будет зависеть лишь от качества программ, предназначенных для нахождения значений специальных функций. Как показывают приведенные выше примеры, даже незначительные на первый взгляд изменения в записи исследуемого ОДУ могут приводить к существенному изменению вида его аналитического решения. При этом поведение решений часто не претерпевает столь же существенных изменений. С другой стороны, при использовании численных методов решения ЗНУ вид исследуемого ОДУ не имеет существенного значения, включая случаи, когда процедура `dsolve` не позволяет получить решение (как в последнем примере). Это обстоятельство является важным достоинством численных методов: они позволяют легко найти решение для широкого класса задач. С учетом нашего большого опыта в области численных вычислений мы можем с уверенностью утверждать, что при исследовании большинства практических задач нахождение их решения в аналитической форме маловероятно и приходится прибегать к использованию численных методов, позволяющих найти численное решение. С другой стороны, аналитические решения, построенные в рассмотренных выше примерах, зависят от произвольной константы `C1`. Поскольку численные методы позволяют найти лишь частное решение, не всегда можно ответить на вопрос о том, как это решение зависит от параметров. Для более ясного понимания этих вопросов численные методы следует использовать в комбинации с аналитическими методами анализа ОДУ, например, вариационными методами или методом возмущений. Другим важным различием между аналитическими и численными решениями является то обстоятельство, что рассмотренные в этой книге численные методы могут использоваться для нахождения решений дифференциальных уравнений, определяемых только гладкими функциями и заданных на конечном интервале. При анализе физических задач часто возникает необходимость во включении в этот интервал сингулярных точек, или в продолжении этого интервала на бесконечность. При анализе подобных уравнений численные методы часто используются в комбинации с асимптотическими методами.

С нашей точки зрения, аналитические и численные методы взаимно дополняют друг друга при решении ОДУ. Эта книга посвящена в основном численным методам, поскольку они просты в использовании и применяются во многих областях человеческой деятельности, но некоторые сложные задачи могут быть решены только аналитическими методами. В последующих главах будут приведены множество примеров использования методов прикладной математики (например, метод асимптотических разложений и метод возмущений) при анализе численных решений ОДУ.

§ 1.2. СУЩЕСТВОВАНИЕ И ЕДИНСТВЕННОСТЬ РЕШЕНИЙ. КОРРЕКТНОСТЬ ПОСТАНОВКИ ЗАДАЧИ РЕШЕНИЯ ОДУ

Прочитав название этого параграфа, читатель возможно подумает, что последующий текст будет представлять собой пример специфической педантичности, которую обычно приписывают всем математикам. Однако вынесенная в заголовок параграфа тема действительно имеет важное практическое значение, поскольку правильно ответив на вопрос о существовании и единственности решения, можно высказать однозначные суждения о возможности нахождения решения рассматриваемого ОДУ, а также о достоверности его численного решения. В этой книге будет приведено множество примеров физически мотивированных задач, которые не имеют решения при определенных значениях параметров. Кроме того, будут рассмотрены задачи, для которых существует множество решений. Очевидно, что у нас должны возникнуть вполне определенные трудности при попытке найти численное решение, которого в действительности не существует. Кроме того, если для рассматриваемого ОДУ существует несколько решений, вообще говоря, неясно, как вычислить именно то решение, которое нас интересует. Математические результаты, гарантирующие существование и единственность решения, хорошо известны, однако при исследовании практических вопросов глубокое понимание существа физического явления имеет очень важное значение.

Установление факта существования и единственности решения для ЗНУ является более простой задачей, чем соответствующая задача для ЗГУ. Для рассматриваемого в этой книге класса ДУЗА указанная задача сводится к аналогичной задаче для ЗНУ. Поэтому в этом параграфе мы сконцентрируем наше внимание на вопросе о существовании и единственности решения для ЗНУ, а случаи ЗГУ и ДУЗА будут более детально исследованы в последующих главах. Подавляющее большинство ЗНУ, возникающих в практических ситуациях, могут быть представлены в виде системы ОДУ первого порядка, разрешенных относительно первой производной

$$\begin{aligned} y_1'(t) &= f_1(t, y_1(t), y_2(t), \ldots, y_d(t)), \\ y_2'(t) &= f_2(t, y_1(t), y_2(t), \ldots, y_d(t)), \\ &\vdots \\ y_d'(t) &= f_n(t, y_1(t), y_2(t), \ldots, y_d(t)). \end{aligned}$$

Для простоты обозначений, перепишем эту систему в векторной форме

$$y'(t) = f(t, y(t)), \tag{1.4}$$

где

$$y(t) = \begin{pmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_d(t) \end{pmatrix}, \qquad f(t, y(t)) = \begin{pmatrix} f_1(t, y(t)) \\ f_2(t, y(t)) \\ \vdots \\ f_d(t, y(t)) \end{pmatrix}.$$

Корректная постановка ЗНУ предполагает, что в начальной точке всем компонентам вектора-решения назначены определенные значения

$$y_1(a) = A_1, \quad y_2(a) = A_2, \quad \ldots, \quad y_d(a) = A_d,$$

или, в векторных обозначениях

$$y(a) = A = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_d \end{pmatrix}. \tag{1.5}$$

В векторной записи система уравнений первого порядка выглядит как скалярное уравнение и на практике многие теоретические результаты, известные для скалярного случая, могут быть применены и к системе уравнений. Следует отметить, что представление модели исследуемой системы в виде системы первого порядка не только удобно с точки зрения применения теоретических результатов, но также и с практической точки зрения. Более детально мы рассмотрим этот вопрос позже.

Выражаясь нестрого, если функция $f(t, y)$ является гладкой для всех значений (t, y) в области R, содержащей начальные данные (a, A), то решение ЗНУ, представленной в виде ОДУ (1.4) с начальными условиями (1.5), существует и единственно. Этот результат может рассматриваться как ответ на вопрос о существовании и единственности решения для большинства возникающих на практике ЗНУ, однако на практике это может оказаться недостаточным и возникает необходимость в том, чтобы точно указать область, где это решение действительно существует. Если решение дифференциального уравнения может быть продолжено до границ области гладкости правой части этого уравнения R, это не означает, что оно существует на заданном интервале $a \leqslant t \leqslant b$, содержащемся в области R. Рассмотрим, например, следующую ЗНУ

$$y' = y^2, \; y(0) = 1.$$

Функция $f(t, y) = y^2$ является гладкой везде, т. е. в области

$$R = \{-\infty < t < \infty, -\infty < y < \infty\},$$

однако единственное решение

$$y(t) = \frac{1}{1 - t}$$

«уходит на бесконечность» при $t \to 1$ и, следовательно, не существует, скажем, на интервале $0 \leqslant t \leqslant 2$, содержащемся в R. Это не противоречит теории, поскольку при $t \to 1$ решение стремится к границе области изменения y-переменной, равной бесконечности. Подобное поведение решения не является исключительным случаем при исследовании физических задач. Таким образом, разумно ставить вопрос о точности численного алгоритма, применяемого для нахождения приближенного решения до момента, когда это решение становится слишком большим и поэтому не может быть вычислено с использованием компьютера. В упражнениях 1.2 и 1.3 рассматриваются аналогичные случаи.

Представление ОДУ и начальных условий соответственно в виде (1.4) и (1.5) является стандартным при исследовании ЗНУ и в параграфе 1.3 мы рассмотрим несколько примеров, показывающих как можно представить исследуемую задачу в этой форме. Свойства существования и единственности решения устанавливаются для этой явной формы представления относительно просто, однако ситуация усложняется при анализе дифференциальных уравнений, представленных в неявной форме

$$F(t, y(t), y'(t)) = 0.$$

С использованием очень простых примеров можно показать, что для подобных уравнений установление свойства существования и единственности решений является непростой задачей. Например, уравнение

$$(y'(t))^2 + 1 = 0$$

не имеет вещественных решений. При исследовании научных и инженерных прикладных задач часто возникает вопрос о том, как решения y системы алгебраических уравнений

$$F(y, \lambda) = 0$$

зависят от (скалярного) параметра λ. Дифференцируя эти уравнения по параметру, получаем систему ОДУ первого порядка

$$\frac{\partial F}{\partial y}\frac{dy}{d\lambda} + \frac{\partial F}{\partial \lambda} = 0.$$

Если при некотором λ_0 можно разрешить алгебраические уравнения $F(y, \lambda_0) = 0$ относительно $y(\lambda_0) = y_0$, то это решение может быть использовано в качестве начального условия ЗНУ, поставленной для представленного выше дифференциального уравнения относительно $y(\lambda)$.Если матрица Якоби

$$J = \frac{\partial F}{\partial y} = \left(\frac{\partial F_i}{\partial y_j}\right)$$

невырождена, можно переписать рассматриваемое ОДУ в стандартной форме

$$\frac{dy}{d\lambda} = -J^{-1}\frac{\partial F}{\partial \lambda}.$$

Однако в случае вырожденности матрицы Якоби установление свойства существования и единственности решения становится сложной задачей. Следует отметить, что подобная ситуация часто возникает при исследовании интересных научных проблем, например в случаях, когда происходит бифуркация решения (изменение в некоторой точке количества решений рассматриваемого дифференциального уравнения). При использовании стандартных численных процедур решения подобных ЗНУ в сингулярной точке (или точке бифуркации) необходимо выполнять дополнительный теоретический анализ поведения решений в окрестности этой точки. В примере 1.1 рассмотрена подобная задача.

В качестве примера системы, в которой возникает бифуркация решений, рассмотрим ОДУ

$$y' = y^2 - \lambda$$

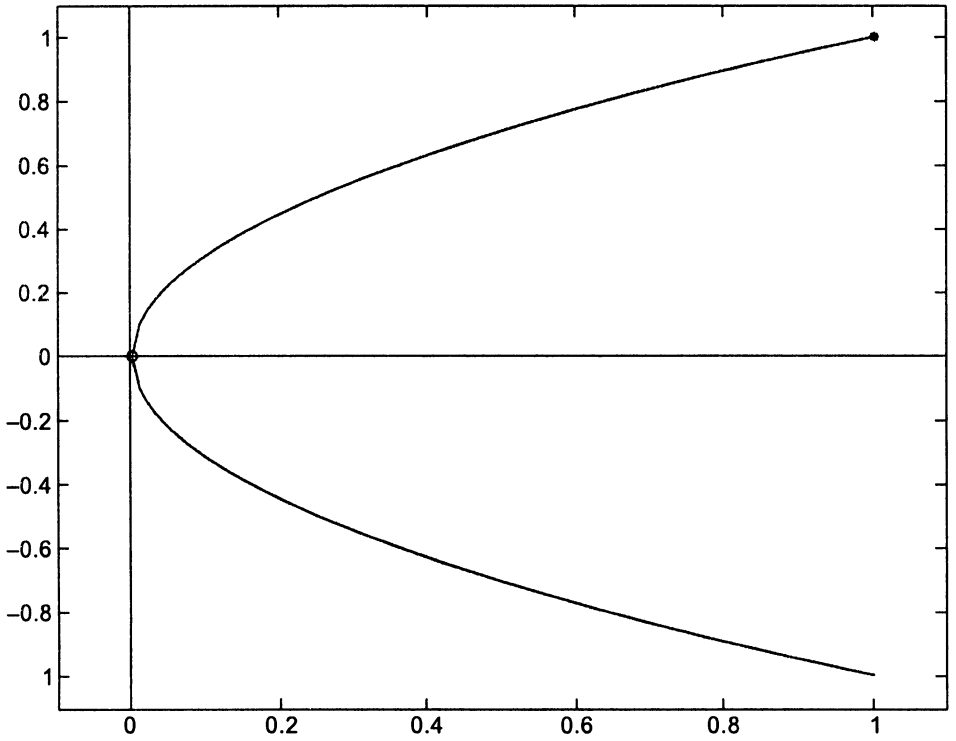

Рис. 1.1. $(0,0)$ — сингулярная точка уравнения $2yy' - 1 = 0$.

и исследуем его стационарные (постоянные) решения, являющиеся решениями алгебраического уравнения

$$0 = y^2 - \lambda \equiv F(y, \lambda).$$

Очевидно, что при $\lambda \geqslant 0$ стационарным решением является $y(\lambda) = \sqrt{\lambda}$. Для того, чтобы исследовать зависимость стационарного решения от параметра λ, можно вычислить его как решение ЗНУ

$$2y\frac{dy}{d\lambda} - 1 = 0, \quad y(1) = 1.$$

При $y \neq 0$ рассматриваемое ОДУ может быть переписано в стандартной форме и решено при значениях λ, убывающих от 1. Однако при $y(\lambda) = 0$, т.е. при $\lambda = 0$, это уравнение сингулярно. Наличие сингулярной точки $(0,0)$ обусловливает возможность существования другого решения, проходящего через эту точку. Действительно, легко показать, что вторым решением является $y(\lambda) = -\sqrt{\lambda}$. С использованием стандартных численных процедур можно выполнить интегрирование рассматриваемого ОДУ при $\lambda = 1$ до момента, когда решение достаточно близко приблизится к началу координат. Очевидно, что в этой точке возникнут проблемы вычислительного характера, поскольку $y'(\lambda) \to \infty$ при $\lambda \to 0$. См. рисунок 1.1.

Для дальнейшего обсуждения численных методов нам необходимо уточнить понятие гладкости функции $f(t,y)$. Этот термин означает, что соответствующая функция является непрерывной в некоторой области R и имеет в этой области непрерывные производные по y настолько большого порядка, насколько это необходимо при дальнейшем анализе. Необходимым условием гладкости функции f в области R является условие Липшица, заключающееся в том, что существует константа L, такая что для любых двух точек (t,u) и (t,v), принадлежащих области R, справедливо неравенство

$$\|f(t,u)-f(t,v)\| \leqslant L\|u-v\|.$$

В скалярном случае из теоремы о среднем значении следует, что

$$f(t,u)-f(t,v)=\frac{\partial f}{\partial y}(t,\zeta)(u-v),$$

т. е. функция $f(t,y)$ удовлетворяет условию Липшица, если величина $\left|\frac{\partial f(t,y)}{\partial y}\right|$ ограничена в области R константой L. Аналогично, если все первые частные производные $\left|\frac{\partial f_s(t,y_1,y_2,\ldots,y_d)}{\partial y_j}\right|$ ограничены в области R, то вектор-функция $f(t,y)$ удовлетворяет условию Липшица в этой области.

Нестрого говоря, задача называется корректно поставленной, если малые изменения начальных данных и параметров приводят к малым изменениям соответствующих решений. Такие задачи называются также хорошо обусловленными по отношению к изменению данных. Это свойство является фундаментальным для физической задачи и численного решения этой задачи. Рассматриваемые в этой книге методы позволяют получить точное решение задачи с начальными данными, малое изменение которых приводит к малому изменению решения этой задачи. Для корректно поставленных задач это означает, что полученное численное решение близко к истинному решению этой задачи. На практике эти аспекты постановки задачи оказываются скрытыми от внимания исследователя, вследствие их зависимости от выбора желаемой точности получаемого решения, а также от точности выполнения арифметических вычислений на используемом компьютере. Рассмотрим примеры, позволяющие выявить потенциальные проблемы.

Маятник представляет собой легкий, жесткий стержень, подвешенный вертикально на шарнире без трения и имеющий на другом свободном конце некоторый груз. При определенном выборе единиц измерения уравнение движения подобной системы может быть представлено в виде ОДУ

$$\theta''+\sin(\theta)=0, \tag{1.6}$$

где $\theta(t)$ — угол отклонения маятника от вертикали. Предположим, что в начальный момент маятник висит вертикально, т. е. начальный угол отклонения равен нулю $\theta(0)=0$. Если начальная скорость $\theta'(0)$ равна нулю, маятник останется неподвижным. Если приложить некоторую силу, начальная скорость $\theta'(0)$ станет ненулевой. При достаточно малой начальной скорости движение маятника будет колебательным. На рисунке 1.2 показаны графики $\theta(t)$ при начальных скоростях $\theta'(0)=-1.9$, 1.5 и 1.9. Однако существует другое решение. Если первоначально приложенная сила достаточно велика, маятник перевернется через верхнюю точку

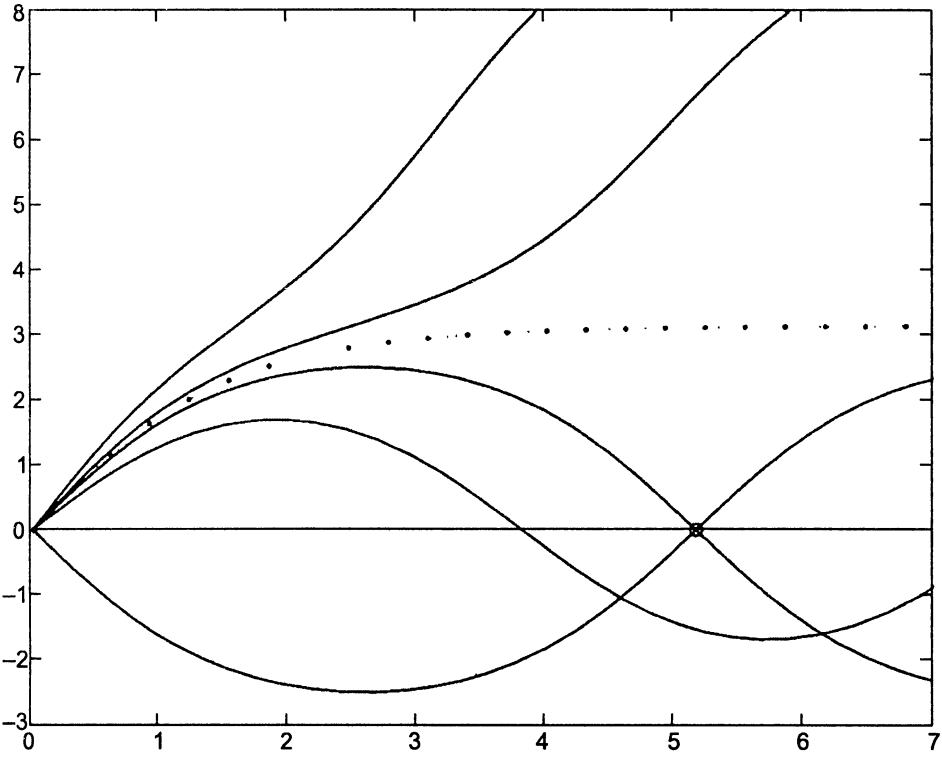

Рис. 1.2. Отклонение угла наклона маятника $\theta(t)$ от вертикали.

и при отсутствии трения в шарнире продолжит свое вращение. Таким образом, при достаточно большой начальной скорости $\theta'(0)$ решение уравнения $\theta(t)$ будет бесконечно возрастать. На рисунке 1.2 показаны два таких решения при начальных скоростях $\theta'(0) = 2.1$ и 2.5. С учетом этих наблюдений можно догадаться, что существует некоторое решение, которое реализуется на практике при некотором специфическом значении начальной скорости и представляет собой переходную форму от колебательного типа решений к бесконечно возрастающим решениям. Это решение изображено на рисунке 1.2 точками. С физической точки зрения это решение соответствует начальной скорости, при которой маятник достигает верхней точки равновесия и останавливается в ней. Очевидно, что это положение равновесия является неустойчивым, поскольку произвольно малое изменение начальной скорости приводит к существенному изменению решения (т. е. поведения) системы. Другими словами, при такой начальной скорости рассматриваемая ЗНУ является некорректно поставленной (плохо обусловленной) на больших интервалах времени. Интересно отметить, что мы можем вычислить начальную скорость, при которой решение уравнения (1.6) становится неустойчивым. Рассматриваемая система является консервативной, т. е. ее энергия

$$E(t) = 0.5(\theta'(t))^2 - \cos(\theta(t))$$

остается с течением времени постоянной. Для доказательства этого факта продифференцируем равенство для $E(t)$ с учетом того, что $\theta(t)$ удовлетворяет ОДУ (1.6). В результате получаем, что производная $E'(t)$ равна нулю при всех t. Поскольку интересующее нас решение удовлетворяет условию $\theta(\infty) = \pi$, разумно предположить, что $\theta'(\infty) = 0$. С учетом того, что начальное значение угла отклонения равно нулю $\theta(0) = 0$, из доказанного выше свойства сохранения энергии следует, что для этого решения справедливо равенство

$$0.5 \times (\theta'(0))^2 - \cos(0) = 0,5 \times 0^2 - \cos(\pi)$$

и, следовательно, $\theta'(0) = 2$. Таким образом, неустойчивое решение уравнения (1.6) возникает при начальных данных $\theta(0) = 0$ и $\theta'(0) = 2$. Остальные решения, изображенные на рисунке 1.2, были вычислены в MATLAB с использованием численной процедуры `ode45` при значении параметра допустимых ошибок вычислений, принятом по умолчанию. Следует отметить, что определяемое этим значением условие, недостаточно жесткое для того, чтобы указанная численная процедура обеспечила получение достаточно точного решения неустойчивой ЗНУ.

Неустойчивое решение вычисляется на практике как решение задачи с граничными условиями, т. е. в рассматриваемом здесь случае как решение ОДУ (1.6) с граничными условиями

$$\theta(0) = 0, \ \theta(\infty) = \pi. \tag{1.7}$$

Если задача моделирования физической системы сформулирована в виде ЗГУ, не всегда ясно, какие при этом следует использовать граничные условия. Как было отмечено выше, с физической точки зрения справедливо равенство $\theta'(\infty) = 0$. Следует ли добавить это условие к условиям (1.7)? Ответ отрицательный, поскольку для решения ЗНУ, представленной уравнением второго порядка, необходимо иметь только два граничных условия и три — это слишком много. Следует ли нам использовать это граничное условие на бесконечности вместо $\theta(\infty) = \pi$? Проблема, с которой мы здесь столкнулись, очевидно заключается в том, что наряду с интересующим нас решением $\theta(t)$, рассматриваемая ЗГУ с граничным условием $\theta'(\infty) = 0$ имеет (по крайней мере) два других решения: $-\theta(t)$ и $\theta(t) \equiv 0$. Изображенное на рисунке 1.2 неустойчивое решение было вычислено в MATLAB с использованием численной процедуры `bvp4c` как решение ЗГУ для (1.6) и (1.7). Эта задача с граничными условиями является корректно поставленной, и поэтому вычисления были выполнены при значении допустимых ошибок вычислений, заданном по умолчанию. С другой стороны, рассматриваемая ЗГУ поставлена на бесконечном интервале, что представляет определенные вычислительные трудности. Все численные процедуры, обсуждаемые в этой книге, предназначены для решения задач, определенных на конечных интервалах. Как мы видели, определение физических задач на бесконечном интервале является вполне естественным и иногда даже необходимым требованием. В подобных ситуациях установление свойств существования и единственности решений, а также корректности постановки задачи, вызывает определенные трудности. Одним из возможных решений этой проблемы является выбор граничного условия в виде конечной величины, но настолько большой, что с учетом физических соображений она может выступать в качестве бесконечности.

Именно этот подход был применен нами в рассматриваемом примере: решение ОДУ было изначально выполнено с граничными условиями

$$\theta(0) = 0,\ \theta(100) = \pi.$$

Оказалось, что столь большой интервал времени $[0, 100]$ в действительности не требуется, поскольку установившееся значение угла θ было почти достигнуто к моменту времени $t = 7$. Для ЗГУ, определяемой уравнением (1.6) и условиями (1.7), полученный выше результат, касающийся величины начальной скорости $\theta'(0) = 2$, можно использовать для оценки требуемой длины интервала интегрирования. Используя процедуру `bvp4c` с принятым по умолчанию значением допустимых ошибок вычислений, было получено численное решение, характеризующееся достаточно большим начальным значением углового коэффициента $\theta'(0) = 1.999979$, что может служить иллюстративным обоснованием рассматриваемого подхода.

Следующий пример показывает, что в зависимости от граничных условий некоторые ЗГУ могут не иметь ни одного решения, или иметь несколько решений. Рассмотрим систему уравнений, описывающую движение артиллерийского снаряда

$$\begin{aligned} y' &= \operatorname{tg}(\phi), \\ v' &= -\frac{g\sin(\phi) + \nu v^2}{v\cos(\phi)}, \\ \phi' &= -\frac{g}{v^2}, \end{aligned} \tag{1.8}$$

где y — высота полета снаряда относительно уровня пушки (высота выстрела), v — скорость движения снаряда, и ϕ — угол (в радианах) наклона траектории по отношению к горизонтальной плоскости. Независимая переменная x равна горизонтальной проекции расстояния снаряда от пушки. Константа ν соответствует модели сопротивления (трения) воздуха и $g = 0.032$ — масштабированная гравитационная постоянная. Эта модель не учитывает трехмерные аспекты движения снаряда, такие как поперечный ветровой снос и вращение снаряда относительно продольной оси. Начальное значение высоты равно нулю $y(0) = 0$, начальная скорость снаряда равна $v(0)$. Стандартная задача состоит в нахождении угла наклона ствола пушки в начальный момент времени $\phi(0)$ (и, следовательно, высоты выстрела), при котором снаряд поразит цель, находящуюся на нулевой высоте и на расстоянии $x = x_{\text{end}}$ от пушки. Таким образом, два граничных условия имеют вид

$$y(0) = y(x_{\text{end}}) = 0, \quad v(0) \text{ задана}.$$

Заметим, что мы указали три граничных условия, поскольку для получения решения исследуемой системы, состоящей из трех уравнений первого порядка, необходимо иметь именно столько граничных условий. Имеет ли эта задача с граничными условиями решение? Разумеется решение не существует при x_{end}, превышающем дальнобойность пушки. С другой стороны, если x_{end} достаточно мало, мы можем надеяться на существование решения, но единственно ли оно? Ответ отрицательный. Действительно, предположим, что цель расположена близко к пушке. В этом случае, мы можем поразить ее, выстрелив почти горизонтально, или направив выстрел по крутой траектории и поразив цель сверху. Таким образом, существует

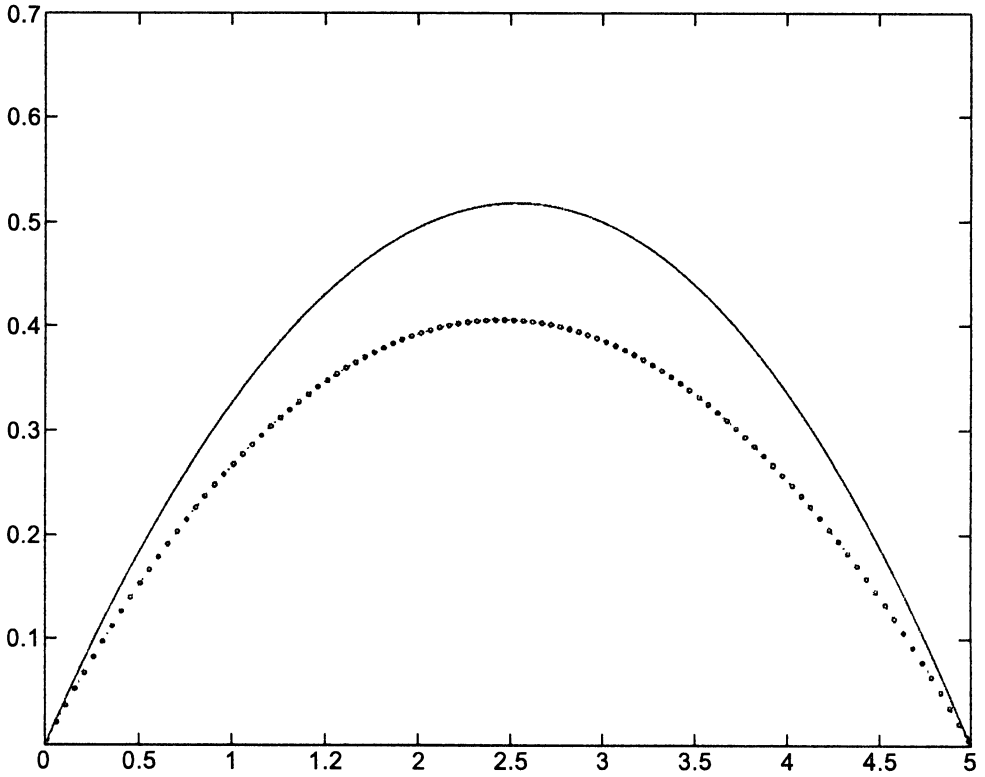

Рис. 1.3. Два способа поразить цель, находящуюся на расстоянии $x_{\text{end}} = 5$ при $v(0) = 0,5$ и $\nu = 0,02$.

(по крайней мере) два решения, которые соответствуют начальным углам $\phi(0) = \phi_{\text{low}} \approx 0$ и $\phi(0) = \phi_{\text{high}} \approx \pi/2$. Можно аналитически показать, что существует ровно два решения. Далее, увеличим x_{end}. В этом случае существует по-прежнему два решения, но чем больше значение x_{end}, тем меньше угол ϕ_{high} и больше угол ϕ_{low}. На рисунке 1.3 показана эта пара траекторий. При дальнейшем увеличении x_{end} будет в конце концов достигнуто максимальное для заданной начальной скорости снаряда значение расстояния, на котором можно поразить цель. При этом значении расстояния существует только одно решение $\phi_{\text{low}} = \phi_{\text{high}}$. Таким образом, может быть найдено критическое значение x_{end}, при котором существует только одно решение. Если x_{end} меньше этого критического значения, существует ровно два решения, если эта величина превышает критическое значение. не существует ни одного решения.

В большинстве физически мотивированных ЗНУ можно сравнительно легко получить результаты о существовании и единственности соответствующих решений. Для ЗГУ также известны соответствующие математические результаты, с использованием которых можно установить факт существования решений, а также определить их количество, но эти результаты настолько специфичны, что редко оказываются применимыми на практике. Для решения подобных задач исследователю

приходится использовать свое понимание физики проблемы для того, чтобы корректно поставить соответствующую задачу и предсказать существование ее решений Точное определение числа решений является чрезвычайно сложной задачей. Кроме того, существует большая вероятность вычисления решения, которое либо является «неверным», либо характеризуется неожиданным поведением. При этом очень важное значение имеет устойчивость численных процедур, используемых при решении ЗНУ, определенных уравнением (1.4) с начальными значениями (1.5). Все рассматриваемые в этой книге численные методы позволяют получить дискретные аппроксимации решения $y_n \approx y(t_n)$ в точках

$$a = t_0 < t_1 < t_2 < \ldots < t_N = b, \tag{1.9}$$

которые выбираются алгоритмом. Интегрирование начинается в заданной начальной точке $y_0 = y(a) = A$ и с использованием значения $y_n \approx y(t_n)$ в момент t_n численный алгоритм вычисляет очередную аппроксимацию в момент $t_{n+1} = t_n + h_n$, где h_n — *шаг численного интегрирования*. Вычисление y_{n+1} осуществляется численным алгоритмом при переходе от t_n к t_{n+1}.

Результат, получаемый с использованием численного алгоритма на очередном шаге, может оказаться неожиданным для исследователя. Решением ЗНУ является локальное решение $u(t)$, удовлетворяющее исследуемому ОДУ

$$u' = f(t, u), \; u(t_n) = y_n. \tag{1.10}$$

При выполнении шага интегрирования численный алгоритм пытается найти такую аппроксимацию y_{n+1}, что *локальная ошибка интегрирования*

$$u(t_{n+1}) - y_{n+1}$$

не превышает заданной пользователем (или заданной по умолчанию) точности получаемого решения, которая влияет на значение *истинной (глобальной) ошибки интегрирования*

$$y(t_{n+1}) - y_{n+1}$$

лишь неявным образом. Для того, чтобы получить представление о процессе возможного распространения ошибки, запишем величину истинной ошибки в момент t_{n+1} в следующем виде

$$y(t_{n+1}) - y_{n+1} = [u(t_{n+1}) - y_{n+1}] + [y(t_{n+1}) - u(t_{n+1})].$$

Первый член в правой части представляет собой локальную ошибку, определяемую численным алгоритмом. Второй член является вычисленной в момент t_{n+1} разностью двух решений ОДУ, которые отличаются друг от друга в момент t_n на величину $y(t_n) - y_n$. Эта разность определяется рассматриваемым ОДУ и, следовательно, используемый численный алгоритм не может ее изменить. Если ЗНУ является устойчивой, т. е. если два решения остаются близкими друг к другу на всем промежутке времени интегрирования, величина истинных ошибок будет сравнима с локальными ошибками. С другой стороны, если рассматриваемая ЗНУ является неустойчивой, т. е. если некоторые из ее решений, начинающиеся вблизи

$y(t)$, быстро удаляются от этой траектории, истинные ошибки могут возрастать, даже если локальные ошибки малы на каждом шаге интегрирования. На рисунке 1.2 показано, что происходит в подобной ситуации. При вычислении неустойчивого решения, изображенного точками, имеют место малые ошибки, что приводит к тому, что это приближенное решение проходит по траекториям соседних решений. Из этого рисунка становится ясно, что локальные решения, начинающиеся вблизи неустойчивого решения, характеризуются большим распространением ошибки, в результате чего накопленная ошибка этих численных решений может оказаться достаточно большой даже в случаях, когда численный алгоритм позволяет получить достаточно точное локальное решение на отдельном шаге интегрирования. Численные ошибки подобного типа представляют собой фундаментальное ограничение всех численных методов, рассматриваемых в этой книге. Если ЗНУ является неустойчивой, то независимо от того, насколько точно вычисляется приближенное решение на отдельном шаге численного интегрирования, на некотором j-ом шаге будет получено значение численного решения y_j, которое будет значительно отличаться от искомого решения $y(t_j)$. Максимальное количество шагов интегрирования, на протяжении которых сохраняется желаемая точность решения, зависит от точности вычисления локальных решений, обеспечиваемой выбранным алгоритмом, а также от степени устойчивости ЗНУ.

Следующий пример иллюстрирует важность того, насколько важно учитывать степень устойчивости рассматриваемой задачи. Решение ОДУ

$$y' = 5(y - t^2), \tag{1.11}$$

с начальным значением $y(0) = 0.08$ имеет вид

$$y(t) = t^2 + 0.4t + 0.08.$$

Вид этой ЗНУ и соответствующее решение не вызывают на первый взгляд опасений, связанных с возможной потерей точности при выполнении численного интегрирования. Тем не менее, рассмотрим общее решение этого ОДУ

$$(t^2 + 0.4t + 0.08) + Ce^{5t}, \tag{1.12}$$

зависящее от произвольной константы интегрирования C. Рассматриваемая ЗНУ является неустойчивой, поскольку решения при $C = C_1$ и $C = C_2$ отличаются друг от друга на величину $(C_1 - C_2)e^{5t}$, которая, как легко видеть, экспоненциально возрастает с течением времени. Для того, чтобы представить соответствующий эффект для численного решения этой ЗНУ, предположим, что на первом шаге интегрирования имела место некоторая малая локальная ошибка, т. е. величина y_1 не была равна точно $y(t_1)$. На очередном шаге выполнения алгоритма численного интегрирования вычисляется аппроксимация локального решения $u(t)$ для рассматриваемого ОДУ при начальном условии $u(t_1) = y_1$. Это решение имеет вид (1.12), где C — малая ненулевая величина, зависящая от начального условия. Предположим, что на всех последующих шагах интегрирования локальные ошибки равны нулю, т. е. $y_n = u(t_n)$ при $n = 2, 3, \ldots$. Тогда истинная ошибка будет равна $y(t_n) - u(t_n) = Ce^{5t_n}$. Вне зависимости от того, насколько была мала локальная

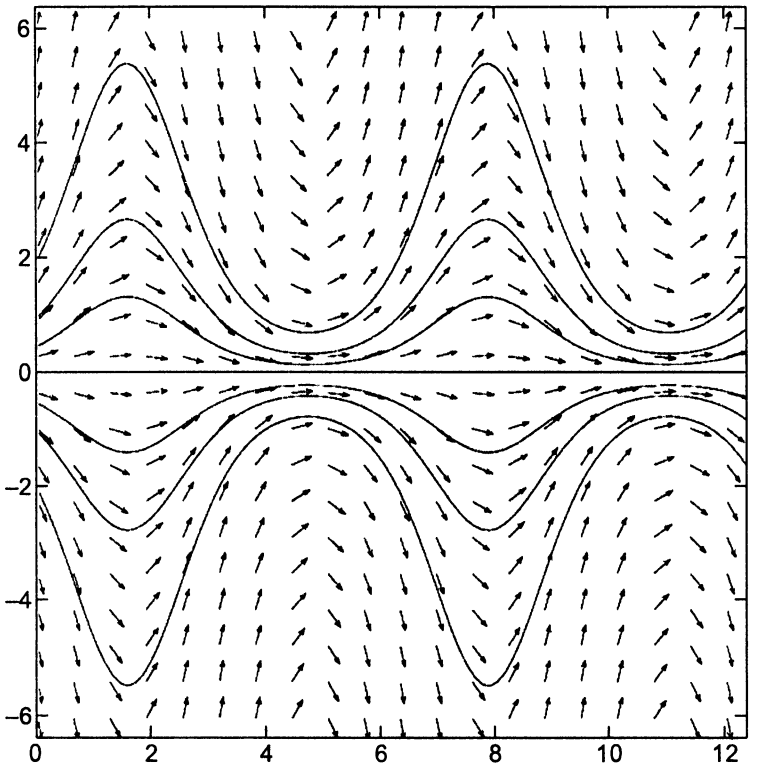

Рис. 1.4. Поле направлений и решения ОДУ $y' = \cos(t)y$.

ошибка на первом шаге интегрирования, экспоненциальный рост истинной ошибки приведет к тому, что точность вычисленного решения y_n станет неприемлемой.

Рассматривая снова рисунок 1.2, можно сказать, что траектории решений при интегрировании в обратном времени (т. е. при движении точек по этим траекториям справа налево) сближаются и, следовательно, показанная точками траектория является устойчивой с вычислительной точки зрения. Этот пример показывает, что устойчивость ЗНУ может зависеть от выбора направления интегрирования. Рассмотрим в этой связи другое ОДУ

$$y' = \cos(t)y. \tag{1.13}$$

Поле направлений и траектории решений этого уравнения изображены на рисунке 1.4. На некоторых интервалах времени решения удаляются друг от друга и, следовательно, численная процедура может быть здесь несколько неустойчивой. На других интервалах времени решения сближаются и процедура интегрирования становится более устойчивой. В случае, когда решения ОДУ характеризуются подобными свойствами, направление, в котором выполняется численное интегрирование, в отличие от предыдущего примера, не играет существенной роли. Рассмотренный пример показывает, что категоричное заявление об устойчивости или неустойчивости определенной ЗНУ было бы иррациональным упрощением проблемы. Кроме

того, возрастание или убывание величины ошибок численного интегрирования на каждом шаге алгоритма может иметь довольно сложную природу. В частности, не следует предполагать, что ошибки всегда накапливаются. В случае интегрирования системы ОДУ одна компонента решения может быть устойчивой по отношению к ошибкам численного интегрирования, в то время как другая может быть неустойчивой. В этом случае крайне затруднительно сделать какие-либо определенные суждения об устойчивости численного алгоритма интегрирования системы в целом и только непосредственные численные эксперименты могут показать, что в действительности происходит с решениями этой системы.

В следующей главе будет рассмотрен метод интегрирования Эйлера, согласно которому численное решение ОДУ $y' = f(t, y)$ находится с использованием алгоритма

$$y_{n+1} = y_n + hf(t_n, y_n). \tag{1.14}$$

Решение уравнения (1.13) с начальным значением $y(0) = 2$ имеет вид

$$y(t) = 2e^{\sin(t)}.$$

Локальным решением $u(t)$ является решение (1.13), проходящее через точку (t_n, y_n)

$$u(t) = y_n e^{(\sin(t) - \sin(t_n))}.$$

На рисунке 1.5 показаны локальная и глобальная ошибки при постоянном шаге интегрирования $h = 0.1$ на промежутке времени от $t = 0$ до $t = 3$. Несмотря на то, что в этом численном алгоритме не реализован контроль величины локальных ошибок, эта величина сильно не изменяется. По определению, локальная и глобальная ошибки равны на первом шаге. Поэтому рост и убывание глобальных ошибок происходит согласованно с изменением устойчивости задачи, подобно тому, как это имело место в примере, результаты численного моделирования которого были представлены на рисунке 1.4.

Обратный анализ ошибок представляет собой важное средство для понимания вопросов, касающихся численных вычислений в области линейной алгебры. Более того, этот анализ позволяет по-новому взглянуть на методы численного интегрирования ОДУ, а также имеет важное значение при алгоритмической реализации этих методов в системе MATLAB. В результате работы всех этих процедур мы получаем кусочно-гладкие приближенные решения $S(t)$ на интервале $[a, b]$, такие что $S(t_n) = y_n$ для всех n. *Невязка* такой аппроксимации определяется равенством

$$r(t) = S'(t) - f(t, S(t)).$$

Рассмотрим $S(t)$ как точное решение возмущенного ОДУ

$$S'(t) = f(t, S(t)) + r(t).$$

В контексте метода обратного анализа ошибки, функция $S(t)$ представляет собой «хорошую» аппроксимацию решения, если она удовлетворяет ОДУ, которое «близко» к исходному, т. е. если невязка $r(t)$ «мала». Это определение «хорошего»

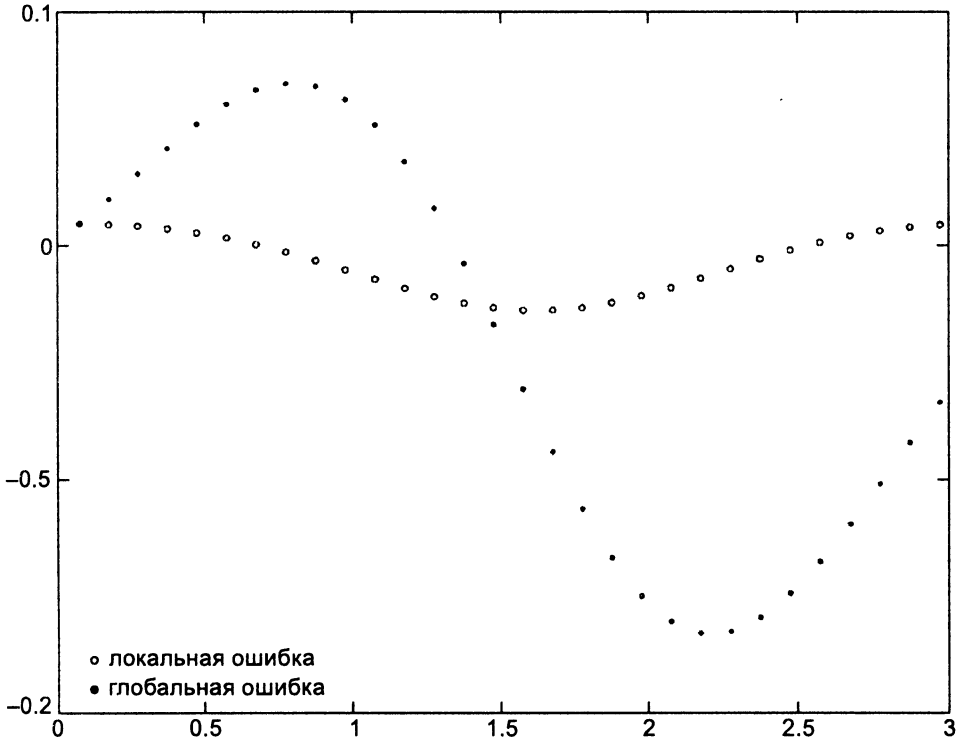

Рис. 1.5. Сравнение локальной и глобальной ошибок.

решения представляется вполне приемлемым и если ЗНУ является корректно поставленной задачей, то из этого определения следует, что в указанном смысле «хорошее» решение «близко» к истинному решению $y(t)$, что соответствует обычному определению хорошей аппроксимации. Таким образом, с использованием численного алгоритма интегрирования находится приближенное решение, характеризующееся малой невязкой. Все используемые в MATLAB численные процедуры решения ЗГУ разработаны с учетом достижения именно этой цели; численные процедуры решения ЗНУ и ДУЗА обеспечивают достижение этой цели опосредованно.

■ УПРАЖНЕНИЕ 1.1

Среди примеров, программный код которых может быть получен в MATLAB с использованием команды `help dsolve`, имеется следующий. Рассматриваемая ЗНУ определяется ОДУ

$$(y')^2 + y^2 = 1,\ y(0) = 0.$$

В примере показывается, что это уравнение может быть легко решено аналитически и соответствующий листинг вывода имеет вид

```
>> y = dsolve('(Dy)^2 + y^2 = 1','y(0) = 0')
y = [  sin(t)]
    [ -sin(t)]
```

Таким образом, с использованием процедуры `dsolve` было получено два решения. Является ли этот результат правильным? Если да, приведите этот результат в соответствие с результатами о существовании и единственности решения ЗНУ, представленными в параграфе 1.2.

■ УПРАЖНЕНИЕ 1.2

Докажите, что функция $f(t,y)$ в правой части ОДУ

$$y' = f(t,y) = +\sqrt{|y|}$$

не удовлетворяет условию Липшица в прямоугольной области $|t| \leqslant 1$, $|y| \leqslant 1$ Покажите, что для этого ОДУ существует несколько решений (более одного), удовлетворяющих начальному условию $y(0) = 0$. Покажите, что $f(t,y)$ удовлетворяют условию Липшица в прямоугольной области $|t| \leqslant 1$, $0 < \alpha \leqslant y \leqslant 1$, т. е. рассматриваемое ОДУ имеет единственное решение при условии, что начальное значение, принадлежит этой прямоугольной области.

■ УПРАЖНЕНИЕ 1.3

Интервал, на котором существует решение некоторой ЗНУ, зависит от начальных условий. Для того, чтобы убедиться в этом, найдите общее решение нижеследующих ОДУ и исследуйте зависимость интервала, на котором существует решение, от начальных условий

$$y' = \frac{1}{(t-1)(t-2)},$$

$$y' = -3y^{4/3}\sin(t).$$

■ УПРАЖНЕНИЕ 1.4

Программа `dfs.m`[1] позволяет вывести в графическом виде поле направлений и решения скалярного ОДУ вида $y' = f(t,y)$. Первым аргументом этой программы является строка, определяющая правую часть $f(t,y)$ рассматриваемого ОДУ. В качестве идентификатора независимой и зависимой переменных должны быть выбраны соответственно t и y. Вторым аргументом является массив $[wL\ wR\ wB\ wT]$, определяющий параметры окна графического вывода: графики решений изображаются для значений $y(t)$ при $wL \leqslant t \leqslant wR$, $wB \leqslant y \leqslant wT$. В первую очередь программа изображает поле направлений. Если после этого пользователь укажет на полученном изображении начальную точку (кликом мышки), программа построит график решения ОДУ, проходящего через эту точку. Клик мышкой вне окна графического вывода прекращает дальнейшее выполнение программы. Например, рисунок 1.4 может быть получен при помощи команды

```
>> dfs('cos(t)*y',[0 12 -6 6]);
```

и последующих кликов левой кнопки мышки в соответствующих точках на полученном графике поля направлений. Программа `dfs.m` была использована нами при

[1] Прим. перев. — Эта и другие цитируемые программы могут быть получены на сайте одного из авторов этой книги: http://www.radford.edu/~thompson/sodeswm/SolvingODESwithMatlab.html

исследовании устойчивости ОДУ (1.11). Графические результаты для этой ЗНУ, исследованной ранее аналитическими методами, были получены при помощи этой программы со вторым аргументом `[0 5 -2 20]`.

■ УПРАЖНЕНИЕ 1.5

Сравните локальную и глобальную ошибки, возникающие при численном решении уравнения (1.11) с начальным значением $y(0) = 0.08$ и постройте график, аналогичный рисунку 1.5. При этом используйте метод Эйлера с постоянным шагом интегрирования $h = 0.1$; в качестве интервала интегрирования возьмите $[0, 2]$. Устойчивость этой задачи была исследована аналитически и численно в упражнении 1.4. С учетом этих результатов обсудите характер изменения глобальных ошибок.

§ 1.3. СТАНДАРТНАЯ ФОРМА ПРЕДСТАВЛЕНИЯ ОДУ

Обыкновенные дифференциальные уравнения могут быть изначально представлены во всевозможных формах. Для решения этих уравнений прежде всего необходимо представить их в той форме, для которой в системе MATLAB существуют соответствующие численные процедуры. Наиболее часто используется рассмотренная в параграфе 1.2 форма записи в стандартной форме системы уравнений первого порядка

$$y' = f(t, y). \tag{1.15}$$

Численные процедуры MATLAB допускают представление исследуемого ОДУ в более общей форме

$$M(t, y)y' = F(t, y), \tag{1.16}$$

где $M(t, y)$ — невырожденная *матрица весовых коэффициентов* (mass matrix). Эти уравнения могут быть переписаны в форме (1.15) с $f(t, y) = M(t, y)^{-1}F(t, y)$, но для некоторых задач форма представления (1.16) является более удобной и эффективной с вычислительной точки зрения. В любом случае, прежде всего необходимо представить исходное ОДУ в виде системы обыкновенных дифференциальных уравнений первого порядка, т. к. оно может иметь порядок более единицы. Стандартным подходом при решении этой задачи является введение новых зависимых переменных. При этом элементам массива новых зависимых переменных следует назначить все зависимые переменные исходной задачи, а также новые переменные, соответствующие всем производным исходных переменных порядка выше первого. Для каждой новой переменной необходимо выписать уравнение, определяющее первую производную этой новой переменной в терминах новых переменных. В итоге мы должны получить новое представление в форме (1.15) (или (1.16)). Все эти манипуляции с переменными и исходными уравнениями легче проиллюстрировать на простых примерах, чем объяснить на словах, как это следует делать. Для того, чтобы представить уравнение маятника (1.6) в стандартной форме, начнем с введения новой переменной $y_1(t) = \theta(t)$. Поскольку в уравнении присутствует вторая производная переменной $\theta(t)$, необходимо ввести

дополнительную переменную $y_2(t) = \theta'(t)$. В этих переменных рассматриваемая система имеет следующий вид

$$y_1'(t) = \theta'(t) = y_2(t),$$
$$y_2'(t) = \theta''(t) = -\sin(\theta(t)) = -\sin(y_1(t)),$$

т. е.

$$y_1' = y_2,$$
$$y_2' = -\sin(y_1).$$

Таким образом, две компоненты вектор-функции $f(t,y)$ в правой части формы представления (1.15) имеют вид $f_1(t,y) = y_2$ и $f_2(t,y) = -\sin(y_1)$. При решении соответствующей ЗНУ для этого ОДУ необходимо задать начальные условия

$$y_1(0) = \theta(0) = 0,$$
$$y_2(0) = \theta'(0).$$

Соответственно, при решении ЗГУ необходимо задать граничные условия

$$y_1(0) = \theta(0) = 0,$$
$$y_1(b) = \theta(b) = \pi.$$

В качестве другого примера рассмотрим уравнения Кеплера, описывающие движение в гравитационном поле двух тел равной массы. При этом центр второго тела выбран в качестве начала координат. При определенном выборе единиц измерения эти уравнения могут быть представлены в следующем виде

$$x'' = -\frac{X}{r^3}, \quad y'' = -\frac{y}{r^3}, \tag{1.17}$$

где $r = \sqrt{x^2 + y^2}$ и $(x(t), y(t))$ — координаты первого тела в системе координат, связанной со вторым телом. При начальных значениях

$$x(0) = 1 - e,\ y(0) = 0,\ x'(0) = 0,\ y'(0) = \sqrt{\frac{1+e}{1-e}} \tag{1.18}$$

можно получить решение (алгебраических) уравнений Кеплера в аналитической форме, которое показывает, что первое тело движется по эллиптической орбите с эксцентриситетом e. Эти уравнения могут быть легко представлены в виде системы ОДУ первого порядка. Одним из возможных выборов новых координат является следующий: $y_1 = x$ и $y_2 = y$. Тогда с учетом того, что в уравнении присутствуют вторые производные, следует ввести дополнительные переменные $y_3 = x'$ и $y_4 = y'$. Легко показать, что во вновь введенных переменных рассматриваемая система имеет вид системы ОДУ первого порядка

$$y_1' = y_3,$$
$$y_2' = y_4,$$
$$y_3' = -\frac{y_1}{r^3},$$
$$y_4' = -\frac{y_2}{r^3},$$

где $r = \sqrt{y_1^2 + y_2^2}$. Соответственно, начальные условия могут быть записаны в следующем виде

$$y_1(0) = 1 - e,\ y_2(0) = 0,\ y_3(0) = 0,\ y_4(0) = \sqrt{\frac{1+e}{1-e}}.$$

Оба эти примера показывают, что механические задачи, основанные на использовании второго закона Ньютона, могут быть описаны системой ОДУ второго порядка. Кроме того, если в системе отсутствует диссипация энергии, правая часть соответствующей системы ОДУ не зависит от первых производных независимых переменных. Уравнения этого типа называются *специальными уравнениями второго порядка*. Для их решения существуют специальные численные процедуры, допускающие представление решаемого уравнения в виде

$$y'' = f(t, y).$$

При этом необходимо задавать начальное состояние системы $y(a)$ и начальную скорость $y'(a)$. Как мы убедились выше, подобные уравнения можно легко представить в форме систем ОДУ первого порядка, но существуют численные методы, в которых эта форма представления эффективно используется при непосредственном интегрировании системы уравнений второго порядка (см., например, Brankin et al. 1989).

Иногда оказывается полезным ввести некоторые дополнительные переменные, которые описывают какие-либо характеристики получаемого решения. В качестве примера подобных задач рассмотрим задачу Штурма–Лиувилля для ОДУ

$$y''(x) + \lambda y(x) = 0$$

с граничными условиями $y(0) = 0$ и $y(2\pi) = 0$. Цель заключается в том, чтобы найти собственное число λ, для которого существует нетривиальное (т. е. не равное тождественно нулю) решение, называемое собственной функцией. В некоторых случаях необходимо находить нормализованное решение, т. е. решение, удовлетворяющее равенству

$$1 = \int_0^{2\pi} y^2(t)dt.$$

Для того, чтобы обеспечить выполнение этого условия нормировки, удобно ввести новую переменную

$$y_3(x) = \int_0^x y^2(t)dt.$$

Тогда в новых переменных $y_1(x) = y(x)$ и $y_2(x) = y'(x)$ исходное уравнение можно представить в виде системы ОДУ первого порядка

$$\begin{aligned} y_1' &= y_2, \\ y_2' &= -\lambda y_1, \\ y_3' &= y_1^2. \end{aligned}$$

По определению третьей переменной в качестве ее начального значения следует выбрать $y_3(0) = 0$. Тогда нас будет интересовать то решение рассматриваемой

системы, которое удовлетворяет условию $y_3(2\pi) = 1$. Таким образом, мы имеем три уравнения, один неизвестный параметр λ и четыре граничных условия

$$y_1(0) = 0,\ y_1(2\pi) = 0,\ y_3(0) = 0,\ y_3(2\pi) = 1.$$

Итак, задача Штурма–Лиувилля может быть решена с использованием введения дополнительной переменной, описывающей желаемые свойства решения. Другим подходом к решению этой задачи является использование представления исходного ОДУ в стандартной форме. Имеющаяся в MATLAB численная процедура `bvp4c` решения ЗГУ позволяет решать задачи с неизвестными параметрами, но эта программа редко используется на практике. В большинстве других численных процедур решения ЗГУ параметр λ заменяется на дополнительную переменную — в рассматриваемом здесь случае $y_4(t)$. Поскольку этот параметр является константой, соответствующая переменная удовлетворяет ОДУ

$$y_4' = 0.$$

Таким образом, в рамках этого подхода мы получаем систему четырех ОДУ первого порядка, которая не включает неизвестный параметр в явном виде

$$\begin{aligned} y_1' &= y_2, \\ y_2' &= -y_4 y_1, \\ y_3' &= y_1^2, \\ y_4' &= 0. \end{aligned}$$

В качестве граничных условий следует выбрать те же, что и в предыдущем случае. В упражнениях 1.8 и 1.9 этот метод используется для представления интегральных ограничений в виде дифференциальных уравнений.

Правильный выбор новых переменных часто имеет ключевое значение для решения задачи. Следующая задача рассматривалась химиком Ф. Сонгом (F.Song) при исследовании коррозии газовых труб с катодной защитой (изложенный ниже результат был представлен в частной беседе). В качестве модели процесса коррозии рассматривалась система ОДУ

$$\begin{aligned} \frac{d^2x}{dz^2} &= \gamma(e^x + \mu_c e^{x\omega_{\mathrm{Fe}}} + \lambda_H e^{x\omega_H} + \lambda_{\mathrm{O}_2} e^{x\omega_{\mathrm{O}_2}}), \\ \frac{d^2 p_{\mathrm{O}_2}}{dz^2} &= \pi p_{\mathrm{O}_2} e^{x\omega_{\mathrm{O}_2}} + \beta p_{\mathrm{O}_2} + \kappa, \end{aligned}$$

определяющая ЗГУ с некоторыми граничными условиями в начале координат и бесконечности. Можно показать, что переменная $p_{\mathrm{O}_2}(z)$ может быть исключена из этих уравнений, в результате чего будет получено уравнение четвертого порядка для единственной переменной $x(z)$. Сведение системы нескольких ОДУ к единственному уравнению высокого порядка часто бывает полезным для последующего анализа, однако для получения соответствующего численного решения необходимо представить это уравнение в виде системы уравнений первого порядка.

В рассматриваемом случае для ОДУ четвертого порядка относительно $x(z)$ можно ввести новые переменные с использованием стандартного подхода, описанного выше

$$y_1 = x,\ y_2 = x',\ y_3 = x'',\ y_4 = x'''.$$

Однако этот путь решения задачи представляется не очень удачным, поскольку при этом теряется информация об эволюции корродирующего агента $p_{O_2}(z)$. Более разумным подходом, является решение задачи в изначальной постановке с введением следующих новых координат

$$w_1 = x,\ w_2 = x',\ w_3 = p_{O_2},\ w_4 = p'_{O_2}.$$

В этом случае упрощается выбор значений требуемой точности вычислений, т. к. эти переменные являются физически обусловленными величинами. Кроме того, задавая требуемую точность вычисления переменной w_3, мы непосредственно определяем точность вычисления интересующей нас переменной p_{O_2}. При решении ЗГУ часто бывает полезным предсказать свойства решения и в рассматриваемом случае это можно сравнительно легко сделать, поскольку все величины имеют физический смысл. В книге Сонга представлена корректная формулировка этой задачи и предложенная автором догадка о поведении решения стала ключевым фактором для успешного нахождения решения этой ЗГУ. Следует отметить, что в практических ситуациях важно не столько найти решение конкретной ЗГУ, сколько решить эту задачу для множества значений параметров, определяющих эту ЗГУ.

■ УПРАЖНЕНИЕ 1.6

Рассмотрим двухточечную ЗГУ, определяемую ОДУ второго порядка

$$(p(x)y')' + q(x)y = r(x)$$

с граничными условиями

$$y(0) = 0,\ p(1)y'(1) = 2.$$

Функция $p(x)$ является дифференцируемой и положительной для всех $x \in [0, 1]$. Представьте это уравнение в виде системы ОДУ первого порядка, введя новые переменные $y_1 = y$ и $y_2 = y'$. В приложениях часто бывает полезным в качестве новой переменной использовать не y', а поток py'. Действительно, одно из граничных условий задано именно для этой переменной. Покажите, что если в качестве новой неизвестной использовать поток, исходное уравнение можно представить в виде системы первого порядка, не дифференцируя $p(x)$.

2 Решение

■ УПРАЖНЕНИЕ 1.7

В работе [Kamke, 1971, стр. 598] рассмотрена ЗНУ, определяемая уравнением

$$y(y'')^2 = e^{2x},\ y(0) = 0, y'(0) = 0,$$

описывающим пространственное распределение заряда в цилиндрическом конденсаторе.

- Представьте это уравнение в виде специального уравнения второго порядка.
- Перепишите это уравнение второго порядка в виде системы уравнений первого порядка.

■ УПРАЖНЕНИЕ 1.8

В работе [Murphy, 1965] были рассмотрены уравнения Фолкнера и Скана (Falkner and Skan), представляющие собой модель ламинарного потока в несжимаемом пограничном слое. При получении результата для более общего случая, когда поверхность обтекаемого тела искривлена, был использован классической метод подобия. Мерфи сформулировал ЗГУ, определяемую уравнением

$$f'''' + (\Omega + f)f''' + \Omega f f'' - (2\beta - 1)[f'f'' + \Omega(f')^2] = 0,$$

на промежутке $0 \leqslant \eta \leqslant b$ и с граничными условиями

$$f(0) = f'(0) = 0,\ f'(b) = e^{-\Omega b},\ f''(b) = -\Omega e^{-\Omega b}.$$

В приведенном выше уравнении Ω — параметр кривизны, β — параметр градиента давления и b — достаточно большая константа, при которой экспоненциальные члены в граничных условиях корректно описывают асимптотическое поведение. Физически обусловленными величинами являются толщина вытеснения (displacement thickness)

$$\Delta^* = \int_0^b [1 - f'(\eta)e^{\Omega\eta}]d\eta$$

и толщина потери импульса (momentum thickness)

$$\theta = \int_0^b f'(\eta)e^{\Omega\eta}[1 - f'(\eta)e^{\Omega\eta}]d\eta.$$

Перепишите уравнение этой ЗГУ в виде системы уравнений первого порядка. Добавьте к этой системе новые уравнения и соответствующие начальные данные так, чтобы величины толщины вытеснения и толщины потери импульса могли быть вычислены вместе с решением $f(\eta)$.

■ УПРАЖНЕНИЕ 1.9

В работе [Caughy, 1970] рассматривается вращательное движение с большой амплитудой гибкой струны. Соответствующая ЗГУ определяется следующим ОДУ

$$\mu'' + \omega^2\left(\frac{1-\alpha^2}{H}\frac{1}{\sqrt{1+\mu^2}} + \alpha^2\right)\mu = 0$$

и начальными условиями

$$\mu'(0) = 0,\ \mu'(1) = 0.$$

Здесь α — некоторая физически обусловленная константа $0 < \alpha < 1$. Поскольку частота вращательного движения ω подлежит определению при решении этой ЗГУ, необходимо задать соответствующее дополнительное граничное условие. В качестве этого условия выбрана амплитуда решения в начале координат

$$\mu(0) = \varepsilon.$$

Нестандартным аспектом этой задачи является то, что константа H определяется на интервале интегрирования в терминах решения $\mu(x)$

$$H = \frac{1}{\alpha^2}\left[1 - (1-\alpha^2)\int_0^1 \frac{dx}{\sqrt{1+\mu^2(x)}}\right].$$

Представьте эту ЗГУ в стандартной форме. Подобно тому, как это было сделано ранее при решении задачи Штурма–Лиувилля, для того, чтобы найти решение с учетом интегрального члена в определении H, можно ввести для этой величины новую переменную $y_3(x)$ и записать соответствующие ОДУ первого порядка и граничное условие. Далее, для величины H следует ввести дополнительную переменную $y_4(x)$. Поскольку H предполагается постоянной, эта последняя переменная удовлетворяет дифференциальному уравнению первого порядка $y_4' = 0$. Решением этого уравнения является константа, значение которой определяется граничным условием, задаваемым с учетом определения H

$$y_4(1) = \frac{1}{\alpha^2}[1 - (1-\alpha^2)y_3(1)].$$

■ УПРАЖНЕНИЕ 1.10

Это упражнение основано на результатах, представленных в книге [Soliman & Srinath, "Continuous and Discrete Signals and Systems", 1998]. Линейная стационарная система (ЛСС) описывается линейным ОДУ с постоянными коэффициентами

$$y^{(N)}(t) + \sum_{i=0}^{N-1} a_i y^{(i)}(t) = \sum_{i=0}^{N} b_i x^{(i)}(t), \tag{1.19}$$

где $x(t)$ — заданный входной сигнал и $y(t)$ — выход системы. Подобные системы могут быть представлены в виде блок-схемы, содержащей лишь блоки усилителей, сумматоров и интеграторов. Соответствующая математическая модель пространства состояний имеет определенные практические преимущества, поскольку система в этом представлении имеет вид системы ОДУ первого порядка, удобной для численного интегрирования. Следует отметить, что подобное представление не единственно и зависит от выбора переменных, но наиболее часто используются две канонические формы. Первую форму можно получить, если в качестве вектора

переменных состояния выбрать $v(t) = (v_1(t), v_2(t), \ldots, v_N(t))^T$. Тогда соответствующая каноническая форма представления системы имеет следующий вид

$$v'(t) = \begin{pmatrix} -a_{N-1} & 1 & 0 & \ldots & 0 \\ -a_{N-2} & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -a_1 & 0 & 0 & \ldots & 1 \\ -a_0 & 0 & 0 & \ldots & 0 \end{pmatrix} v(t) + \begin{pmatrix} b_{N-1} - a_{N_1} b_N \\ b_{N-2} - a_{N_2} b_N \\ \vdots \\ b_1 - a_1 b_N \\ b_0 - a_0 b_N \end{pmatrix} x(t).$$

Выход системы $y(t)$ определяется равенством

$$y(t) = (1, 0, \ldots, 0)^T v(t) + b_N x(t).$$

Докажите, что решение ОДУ (1.19) эквивалентно решению этой системы ОДУ первого порядка с учетом того, что коэффициенты являются константами.

Указание: Используя тождество

$$y(t) = v_1(t) + b_N x(t),$$

перепишите эту систему в виде

$$v_i'(t) = (b_{N-i} x(t) - a_{N-i} y(t)) + v_{i+1}(t)$$

для всех $i < N$. Продифференцируйте это равенство для $v_1'(t)$ и, используя уравнение для $v_2'(t)$, получите уравнение для $v_1''(t)$ в терминах $v_3(t)$. Повторите эту процедуру рекурсивно. В результате вы получите выражение для $v_1^{(N)}(t)$, которое следует приравнять $(y(t) - b_N x(t))^{(N)}$. Сравните полученное уравнение с ОДУ (1.19).

Вторая каноническая форма представления ЛСС имеет следующий вид

$$v'(t) = \begin{pmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \\ -a_0 & -a_1 & -a_2 & \ldots & -a_{N-1} \end{pmatrix} v(t) + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} x(t).$$

Выход системы $y(t)$ в этом случае определяется более сложным равенством

$$y(t) = [(b_0 - a_0 b_N),\ (b_1 - a_1 b_N),\ \ldots,\ (b_{N-1} - a_{N-1} b_N)]^T v(t) + b_N x(t).$$

Докажите, что решение ОДУ (1.19) эквивалентно решению системы ОДУ первого порядка, представленной выше.

Указание: Определите функцию $w(t)$ как решение ОДУ

$$w^{(N)}(t) + \sum_{j=0}^{N-1} a_j w^{(j)}(t) = x(t)$$

и покажите с использованием соответствующей подстановки, что функция

$$y(t) = \sum_{i=0}^{N} b_i w^{(i)}(t)$$

удовлетворяет ОДУ (1.19). Наконец, получите стандартную форму представления системы ОДУ первого порядка для функции $w(t)$.

Интересно отметить, что производные $x^{(i)}(t)$ не входят ни в одну из представленных канонических форм представления. Покажите, что эти производные будут необходимы при нахождении начальных условий $v_i(0)$, соответствующих начальным условиям для первоначальных переменных $y^{(i)}(0)$ и $x^{(i)}(0)$.

§ 1.4. КОНТРОЛЬ ОШИБОК ВЫЧИСЛЕНИЙ

При использовании численных процедур интегрирования ОДУ необходимо задать точность вычислений; чем больше требуемая точность, тем больше вычислительной работы выполнит компьютер. При использовании в MATLAB процедур интегрирования задается скалярная величина допустимой относительной ошибки вычислений re и вектор значений допустимой абсолютной ошибки вычислений ae. Выходными данными этих процедур являются векторы $y_n = (y_{n,i})$, являющиеся аппроксимациями решения $y(t_n) = (y_i(t_n))$ в точках (1.9). Говоря нестрого, в каждой такой точке вычисляется аппроксимация решения, такая что каждая компонента этого решения удовлетворяет неравенству

$$|y_i(t_n) - y_{n,i}| \leqslant re|y_i(t_n)| + ae_i. \tag{1.20}$$

Во всех численных процедурах решения ЗНУ используются аналогичные критерии контроля ошибки вычислений. При этом для всех компонент вектора решения часто используется единое скалярное значение абсолютной ошибки вычислений, что иногда бывает удобно пользователям. Кроме того, в этих процедурах предусмотрено назначение этим величинам значений по умолчанию, которые равны 10^{-3} и 10^{-6} соответственно для допустимых относительной и абсолютной ошибок вычислений. При выполнении научных вычислений обычно используется большее значение относительной ошибки вычислений 10^{-5}.

В отношении выбора векторов относительной и абсолютной ошибки вычислений в работе [Brenan, Campbell, & Petzold, 1996, стр. 131] сказано: «Нельзя не отметить важность тщательного выбора значений этих величин вследствие тесной связи этого вопроса с различием масштаба компонентов решения. В частности, для задач с решениями, компоненты которых характеризуются существенно различными масштабами, рекомендуется использовать векторные величины допустимых ошибок вычислений для того, чтобы имелась возможность задавать различные их значения для соответствующих компонент решения. В случае возникновения затруднений при выборе значений компонент векторов относительной и абсолютной ошибок вычислений RTOL и ATOL рекомендуется использовать следующее правило. Пусть m — число значащих цифр, необходимых в i-й компоненте решения y_i. Тогда компоненты вектора RTOL_i определяются равенством $\mathrm{RTOL}_i = 10^{-(m+1)}$; значения компонент вектора ATOL_i должны быть выбраны так, чтобы величины $|y_i|$ были ничтожно малыми.»

Этот параграф посвящен различным вопросам, связанным с выбором величины допустимых ошибок вычислений. Представленное обсуждение поможет читателю понять существо предложенного выше правила.

Неравенство (1.20) определяет комбинированный критерий контроля ошибки вычислений. Случай, когда для всех i выполнено $ae_i = 0$, соответствует контролю только относительной ошибки (относительный критерий); при $re = 0$ этот критерий сводится к контролю только абсолютной ошибки (абсолютный критерий). При контроле только относительной или только абсолютной ошибки более четко выявляются роли этих критериев в вычислительном процессе, а также проблемы, связанные с использованием соответствующих критериев. Рассмотрим в первую очередь вопрос о контроле относительной ошибки вычислений. Соответствующий критерий для каждой компоненты вектора решения имеет вид следующего неравенства

$$\left| \frac{y_i(t_n) - y_{n,i}}{y_i(t_n)} \right| \leqslant re.$$

При этом возникают две существенные проблемы. Первая из них заключается в том, что контроль лишь относительной ошибки является неадекватным критерием в ситуациях, когда решение, т. е. знаменатель $y_i(t_n)$ этой дроби, может обращаться в ноль. С фундаментальной точки зрения указанную проблему можно сформулировать в виде следующего вопроса: как определить функцию относительной ошибки, если $y_i(t)$ может обращаться в ноль в некоторой изолированной точке $t = t^*$? В численных процедурах интегрирования относительная ошибка обычно сравнивается не с $|y_i(t_n)|$ из (1.20), а с некоторой величиной, представляющей собой меру $y_i(t)$ вблизи t_n. Это подход представляется вполне разумным и эффективным, но он неприменим в ситуации, когда одна из компонент решения $y_i(t)$ равна нулю на некотором промежутке в окрестности t_n. Таким образом, эти численные процедуры должны во-первых распознавать ситуации, когда критерий контроля относительной ошибки перестает быть адекватным даже в некотором расширенном смысле и во-вторых прерывать вычислительный процесс с уведомлением об этом пользователя. Возникновение этой проблемы можно предотвратить, если использовать комбинированный критерий с ненулевой абсолютной ошибкой. Для обеспечения робастности алгоритмов численных процедур, в том числе процедур, входящих в состав системы MATLAB, предполагается, что допустимой абсолютной ошибке вычислений назначаются положительные значения.

Прежде чем перейти к рассмотрению второй проблемы, связанной с критерием контроля относительной ошибки, необходимо сделать некоторые комментарии, касающиеся выполнения арифметических операций на компьютере. Языки программирования, например Fortran 77 и C, допускают выполнение вычислений с нормальной и двойной точностью, что соответствует 7 и 16 десятичным цифрам в представлении числа. В системе MATLAB вычисления выполняются только с двойной точностью. Практика показывает, что при численном решении ЗНУ следует использовать двойную точность. Число с плавающей точкой можно представить лишь с точностью до ошибки округления, определяемой точностью вычислений, доступной при использовании конкретного компьютера. В MATLAB эта величина обозначается идентификатором eps и в случае использования PC она равна $2.2204 \cdot 10^{-16}$, что по стандарту IEEE-754 соответствует двойной точности

арифметических операций, выполняемых на практически всех используемых в настоящее время компьютерах. Далее в этой книге мы будем предполагать, что величина ошибки округления равна указанной выше величине.

Нестрого говоря, относительная точность вычислений определяет количество верных цифр в значении искомого решения. Бессмысленно пытаться обеспечить точность вычислений, большую чем доступная на данном компьютере точность представления числа с плавающей точкой. Иными словами, бесполезно задавать значение re меньше ошибки округления. Разумеется, значение допустимой ошибки, близкое к значению ошибки округления, также является слишком малой величиной, поскольку выполнение вычислений с ограниченной точностью неизбежно влияет на точность, обеспечиваемую соответствующей численной процедурой. Поэтому во всех используемых в MATLAB численных алгоритмах предполагается, что величина re должна быть больше некоторого значения, кратного величине eps. Соответствующий множитель задается в алгоритме каждой численной процедуры интегрирования. В идеальной ситуации алгоритм должен прекращать свою работу и извещать об этом пользователя в тех случаях, когда затребована недостижимая точность вычислений. К сожалению, в большинстве практических ситуаций это не происходит. Вероятнее всего при экспериментах с параметрами своей программы вы будете наблюдать следующую картину. При уменьшении значения допустимой ошибки вычислений процедура численного интегрирования становится вычислительно затратной, а точность получаемых результатов убывает. При этом используемая вами численная процедура не выдает каких-либо сообщений, свидетельствующих о нарушении критерия контроля ошибки.

Обратимся к рассмотрению критерия контроля абсолютной ошибки, который можно записать для каждой компоненты решения в виде следующего неравенства

$$|y_i(t_n) - y_{n,i}| \leqslant ae_i.$$

Основной проблемой, возникающей при реализации этого критерия, является то, что при этом необходимо получать достаточно достоверные оценки величин компонент решения. В случае, если эти оценки оказываются неверными возникают определенные трудности. Рассмотрим, например, ситуацию, когда абсолютная величина компоненты решения в действительности больше, чем ожидалось. Переписав приведенный выше критерий контроля абсолютной ошибки в виде

$$\left|\frac{y_i(t_n) - y_{n,i}}{y_i(t_n)}\right| \leqslant \frac{ae_i}{|y_i(t_n)|},$$

можно заметить, что абсолютная точность ae_i вычисления компоненты $y_i(t)$ соответствует относительной точности $ae_i/|y_i(t_n)|$ для той же компоненты. Если величина $|y_i(t_n)|$ достаточно велика, вполне приемлемое на первый взгляд значение абсолютной точности может соответствовать требованию обеспечения относительной точности вычислений, превышающей ошибку округления. Как мы видели выше, подобная точность недостижима на практике. Возникновение подобной ситуации можно предотвратить, если использовать комбинированный критерий с ненулевой относительной ошибкой. Для обеспечения робастности используемых в MATLAB численных процедур предполагается, что допустимая относительная ошибка вычислений больше ошибки округления в несколько раз.

Другая проблема, связанная с использованием критерия контроля абсолютной ошибки, возникает в случаях, когда компонента решения становится существенно меньше значения абсолютной точности. В первую очередь необходимо прояснить смысл критерия контроля абсолютной ошибки для компоненты решения в подобной ситуации. Например, если $|y_i(t_n)| < 0.5ae_i$, то *любая* аппроксимация решения $y_{n,i}$, для которой выполнено неравенство $|y_{n,i}| < 0.5ae_i$, удовлетворяет этому критерию, но легко понять, что приемлемая в этом смысле аппроксимация может не содержать требуемое количество верных цифр. Часто полагают, что решение всегда необходимо вычислять с требуемой точностью, но в математических моделях реальных физических процессов могут использоваться переменные, которые оказывают бесконечно малое воздействие на общую эволюцию системы, если их величина становится меньше некоторых критических значений. В этой связи можно предположить, что после этого момента о точности вычисления этих переменных можно забыть, но это делать не следует, т. к. вполне возможно, что позднее их значение снова возрастет и эффект от их изменения снова станет заметен. Если абсолютная величина компоненты решения становится меньше соответствующей этой компоненте требуемой абсолютной точности вычислений, следует изменить это значение и выполнить повторные вычисления. Интересно отметить, что в некоторых случаях удается найти решение с «малой» по абсолютной величине компонентой, содержащей некоторое количество верных цифр, несмотря на то, что соответствующее требование на точность вычислений не было определено в виде заданного значения абсолютной точности. Это может быть объяснено тем, что численная процедура вычислила эту компоненту с точностью, которая оказалась необходимой для обеспечения точности, заданной для другой компоненты решения, зависящей от первой компоненты. Другим возможным объяснением подобной ситуации является то, что вычисления выполнялись при достаточно малом шаге интегрирования, который был уменьшен этой процедурой в целях обеспечения требуемой точности для вычисления другой быстро изменяющейся компоненты решения. В этом случае шаг интегрирования является излишне малым для остальных компонент решения, но они вынужденно вычисляются с той точностью, которая необходима для вычисления сравнительно быстрой компоненты.

Первый пример, который иллюстрирует обсужденные выше вопросы, рассмотрен в работе [Lapidus, Aiken, & Liu, 1973]. Протонный перенос в водород-водородной связи описывается системой ОДУ

$$\begin{aligned} x_1' &= -k_1 x_1 + k_2 y, \\ x_2' &= -k_4 x_2 + k_3 y, \\ y' &= k_1 x_1 + k_4 x_2 - (k_1 + k_3) y. \end{aligned} \tag{1.21}$$

Эта система должна быть решена с начальными данными

$$x_1(0) = 0,\ x_2(0) = 1,\ y(0) = 0$$

на интервале $0 \leqslant t \leqslant 8 \cdot 10^5$. Коэффициенты определяются следующими равенствами

$$\begin{aligned} k_1 &= 8.4303270 \cdot 10^{-10},\ \ k_2 = 2.9002673 \cdot 10^{11}, \\ k_3 &= 2.4603642 \cdot 10^{10},\ \ \ k_4 = 8.7600580 \cdot 10^{-6}. \end{aligned}$$

Подобные задачи называются *жесткими*. Решение этого уравнения может быть получено в MATLAB с использованием численной процедуры решения ЗНУ `ode15s` при значениях абсолютной и относительной точности, заданных по умолчанию, однако в ходе численных экспериментов было установлено, что компонента $y(t)$ очень быстро становится меньше значения допустимой абсолютной ошибки 10^{-6}, принятой по умолчанию. Тем не менее, указанная компонента была вычислена достаточно точно, что позволило предварительно оценить ее величину. После того, как стало ясно насколько мала эта компонента решения, мы уменьшили допустимую абсолютную ошибку до 10^{-20} и получили решение, изображенное на рисунках 1 6 и 1.7. Применяемые в MATLAB процедуры численного интегрирования позволяют построить траектории всех компонент решения на одном графике, что очень удобно при проведении предварительных исследований. Если одна из компонент решения становится слишком малой по сравнению с другими, вы можете построить для нее отдельный график в соответствующем масштабе. Кроме того, подобное поведение решения исследуемой системы может навести вас на мысль о том, что при последующих запусках вашей программы следует увеличить значения требуемой точности вычислений.

При моделировании химических реакций концентрации реагентов могут стать настолько малыми, что с этого момента эти величины перестают оказывать какое-либо воздействие на исследуемый химический процесс. В подобных ситуациях в качестве абсолютной точности вычислений разумно использовать соответствующие критические значения концентраций. Убывающие до нуля значения концентраций $y_i(t)$ по своей физической природе являются положительными величинами, но в процессе численного моделирования соответствующие аппроксимации очень малой компоненты решения могут стать отрицательными $y_{n,i} < 0$. Как мы видели ранее, критерий контроля ошибок допускает подобную ситуацию и этот эффект действительно имеет место при выполнении вычислений на практике. В общем случае малые отрицательные значения концентраций не должны вызывать беспокойства экспериментатора, но существуют модели, которые перестают быть устойчивыми в подобных ситуациях и соответствующее решение уходит на бесконечность. Следует иметь в виду, что величины, столь малые, что не имеют физической значимости, могут разрушить численное решение. В работе [Robertson, 1966] рассмотрен пример, который может служить в качестве теста для численных процедур решения жестких ЗНУ. Химическая реакция описывается системой ОДУ

$$\begin{aligned} y_1' &= -0.04y_1 + 10^4 y_2 y_3, \\ y_2' &= 0.04y_1 - 10^4 y_2 y_3 - 3 \cdot 10^7 y_2^2, \\ y_3' &= 3 \cdot 10^7 y_2^2 \end{aligned} \tag{1.22}$$

с начальными условиями

$$\begin{pmatrix} y_1(0) \\ y_2(0) \\ y_3(0) \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

Нетрудно показать, что для всех $t > 0$ компоненты решения неотрицательны и их сумма равна 1. Это может служить примером линейного закона сохранения, который будет рассмотрен в следующем параграфе более подробно.

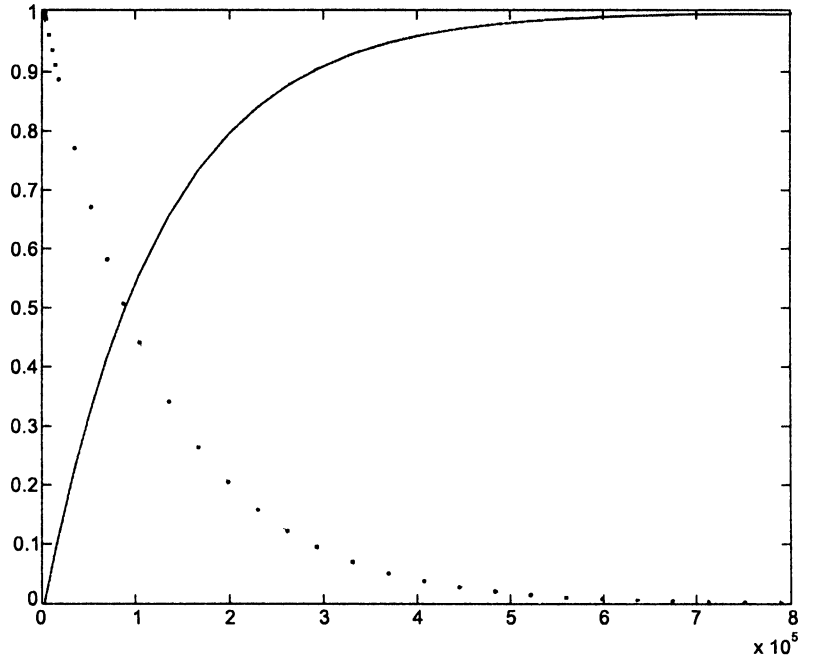

Рис. 1.6. Графики компонент решения $x_1(t)$ и $x_2(t)$ в задаче о протонном переносе.

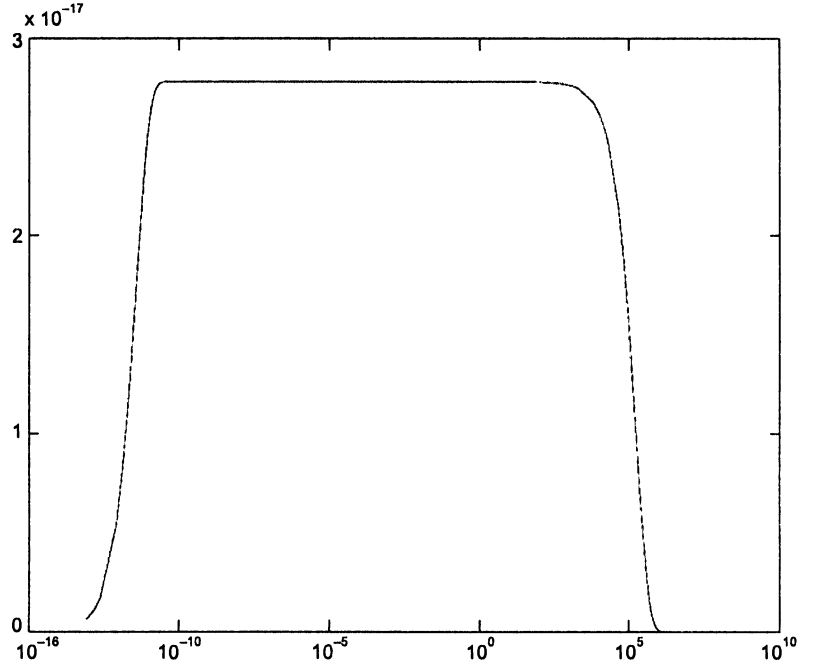

Рис. 1.7. График компоненты решения $y(t)$ в задаче о протонном переносе, выведенный с использованием функции `semilogx`.

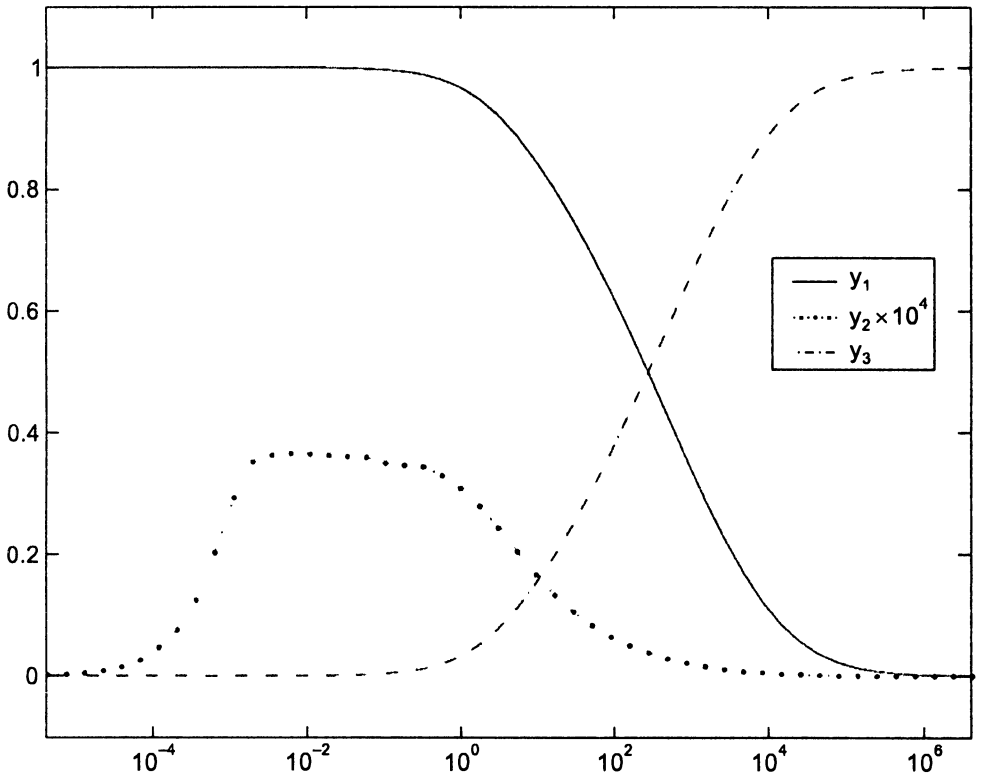

Рис. 1.8. Решение задачи Робертсона, представленное с использованием процедуры `semilogx`.

Демонстрационная программа `hb1ode`, входящая в состав системы MATLAB, позволяет выполнить интегрирование этой системы с использованием численной процедуры `ode15s` на промежутке от $t = 0$ до момента $t = 4 \cdot 10^6$, когда система практически достигает установившегося режима. Несколько модифицированный вывод этой программы представлен на рисунке 1.8. В работе [Hindmarsh & Byrne, 1976] решение этой задачи использовалось в качестве иллюстрации работоспособности программы `EPISODE`, предназначенной для решения жестких ЗНУ. Было показано, что при умеренном значении абсолютной точности 10^{-6} в процессе интегрирования системы значения концентраций принимают физически необоснованные неотрицательные значения, что приводит к тому, что соответствующее решение начинает быстро возрастать и становится в конечном счете совершенно неприемлемым. Часть соответствующих численных значений представлена в таблице 1.1. Подчеркнем, что неудовлетворительная работа программы обусловлена особенностями задачи, но не неудачной программной реализацией численного алгоритма. Аналогичные результаты могут быть получены и с использованием других численных процедур интегрирования, например `ode15s`. Более подробно эта задача рассмотрена в работах [Hindmarsh & Byrne, 1976] и [Shampine, 1994]. Другие задачи со сходными особенностями рассмотрены в упражнениях 1.12 и 1.13.

Таблица 1.1: Установившееся решение задачи Робертсона, вычисленное с использованием программы EPISODE

t	y_1	y_2	y_3
4e5	4.9394e−03	1.9854e−08	9.9506e−01
4e7	3.2146e−05	1.2859e−10	9.9997e−01
4e9	−1.8616e+06	−4.0000e−06	1.8616e+06

Несмотря на естественный соблазн задать относительно малую допустимую ошибку вычислений, этого делать не следует, т. к. чем больше требуемая точность, тем больше вычислительные затраты. Выбор большой точности особенно заманчив, если данные в рассматриваемой задаче известны лишь с точностью до одной или двух цифр. (Нам даже приходилось решать задачи, в которых порядок измеряемых величин был заранее неизвестен.) Тем не менее, задание слишком малой допустимой ошибки вычислений представляется слишком тривиальным подходом к решению задачи и в отдельных случаях это может быть чревато возникновением различных проблем. С другой стороны, большинство численных алгоритмов работоспособны лишь при достаточно малом шаге интегрирования. Если мы не зададим достаточно большую точность вычислений, численная процедура может выбрать шаг слишком большим, вследствие чего будут получены не вполне адекватные результаты. Хорошо реализованные численные алгоритмы могут распознать ситуацию, когда следует уменьшить шаг интегрирования и без участия пользователя уменьшают заданное им значение допустимой ошибки вычислений. Как было показано в параграфе 1.2, численные процедуры обеспечивают контроль локальных ошибок вычислений, а погрешности вычисления самого решения $y(t)$ контролируются лишь неявным образом. Численные алгоритмы реализованы таким образом, чтобы локальные ошибки не превышали заданных допустимых значений. Особенности алгоритма «подстройки» этих величин зависят от используемого численного метода, но для типичных ЗНУ этот алгоритм гарантирует, что ошибка при вычислении решения $y(t)$ будет меньше (или сравнима), чем заданная точность вычислений. Однако, если исследуемая ЗНУ нетипична и демонстрирует некоторую неустойчивость или высокочастотное поведение на рассматриваемом интервале, всегда следует быть очень осмотрительным при выборе точности вычислений, т. к. задание слишком большой допустимой ошибки может привести к обескураживающим результатам. Вычисленное решение может оказаться физически нереалистичным, или физически реалистичным но неправильным, или вычислительный процесс может стать неустойчивым и прерваться. При анализе подобных ситуаций следует принимать во внимание то обстоятельство, что функционирование численной процедуры определяется входными данными — процедура обеспечивает такую локальную ошибку, которая не превышает заданное пользователем допустимое значение. Получение неудовлетворительных результатов — это следствие неустойчивости ЗНУ, но не ошибок программной реализации численного алгоритма интегрирования. На рисунке 1.9 в параграфе 1.5 показаны результаты, которые характерны для подобных ситуаций. Показанная точками траектория — это

орбита, по которой вращается одно тело вокруг другого. Соответствующее решение было вычислено при значении допустимой ошибки, принятом по умолчанию. Эти значения позволяют получить вполне адекватные результаты при решении типичных задач, но в рассматриваемом случае полученная траектория орбиты не является корректной даже по своим качественным характеристикам. Траектория, изображенная непрерывной линией, была получена при более сильных требованиях на точность вычислений и она является физически адекватной. Этот пример показывает, что задание слишком малой точности вычислений может привести к абсолютно некорректным результатам.

При решении новой ЗНУ рекомендуется выполнить несколько экспериментов при различных значениях допустимой ошибки. Для правильного выбора этого значения необходимо проанализировать полученные решения и удостовериться, что используемый критерий контроля ошибки вычислений позволяет получить адекватные решения рассматриваемой задачи. Кроме того, следует попытаться уменьшить значения допустимой ошибки для того, чтобы убедиться в том, что заданная точность вычислений обеспечивает получение физически корректных результатов.

■ УПРАЖНЕНИЕ 1.11

Для простоты использования некоторых численных процедур предусмотрена возможность задания единой точности вычислений τ для всех компонент решения. Например, в процедуре DVERK [Hull, Enright, & Jackson, 1975] на каждом шаге используется следующий критерий

$$|y_i(t_n) - y_{n,i}| \leqslant \tau \max(1, |y_i(t_n)|).$$

В процедуре MIRKDC [Enright & Muir, 1996] соответствующий критерий имеет вид

$$|y_i(t_n) - y_{n,i}| \leqslant \tau(1 + |y_i(t_n)|).$$

При этом утверждается, что этот подход практически эквивалентен критерию контроля ошибки (1.20) с $re = \tau$ и $ae_i = \tau$ для каждого i. Пользователи часто сталкиваются с некоторыми трудностями при использовании процедур с подобными критериями контроля ошибки, т. к. они не понимают, что задаваемые ими значения допустимой абсолютной ошибки не соответствуют той задаче, которую они пытаются решить.

■ УПРАЖНЕНИЕ 1.12

Решение уравнения

$$y' = f(t, y) = \sqrt{1 - y^2}, \; y(0) = 0$$

имеет вид $\sin(t)$. Почему при численном нахождении этого решения возникают проблемы при приближении к точке, соответствующей моменту времени $t = 0.5\pi$? Указанные проблемы связаны, во-первых, с используемым критерием контроля ошибки и, во-вторых, с единственностью решения.

■ УПРАЖНЕНИЕ 1.13

При численном решении в системе MATLAB дифференциального уравнения

$$y' = \left(\frac{2\ln(y) + 8}{t} - 5\right) y, \; y(1) = 1$$

вычисляются аппроксимации решения $y(t)$, которые принимают комплексные значения при «больших» t. Численные процедуры других программных систем для математических расчетов могут вообще не справиться с решением этой задачи. Чем это обусловлено? Для ответа на этот вопрос полезно использовать аналитическое решение этого уравнения

$$y(t) = e^{-t^2+5t-4}.$$

§ 1.5. КАЧЕСТВЕННЫЕ СВОЙСТВА РЕШЕНИЙ

В рассмотренных выше примерах мы использовали аналитические решения, которые обладали некоторыми специфическими свойствами качественного характера, обусловленными соответствующими ОДУ. Часто полагают, что численные решения также должны обладать этими свойствами, но в большинстве случаев это не так. Стандартные численные методы позволяют лишь аппроксимировать соответствующее поведение аналитического решения. Существуют модификации стандартных численных методов, которые позволяют получать решения с сохранением их качественных свойств в тех ситуациях, когда не модифицированные алгоритмы оказываются бесполезными, но в этой книге мы не будем рассматривать столь специфические аспекты численного интегрирования. Детальное исследование этих вопросов представлено в [Sanz-Serna & Calvo, 1994], [Shampine, 1986], [Stuart & Humphries, 1996].

Рассмотрение качественных характеристик решений дифференциальных уравнений мы начнем с обсуждения свойства, которое оказывается выполненным для численных решений, получаемых с использованием практически всех стандартных методов интегрирования. Если существует постоянный вектор-столбец c, такой что $c^T f(t,y) \equiv 0$, то решение системы ОДУ

$$y' = f(t,y),\ y(a) = A$$

удовлетворяет *линейному закону сохранения*

$$c^T y(t) \equiv c^T A.$$

Этот результат следует из равенства

$$\frac{d}{dt}(c^T y(t)) = c^T y'(t) = c^T f(t, y(t)) \equiv 0,$$

т.е. величина $c^T y(t)$ является постоянной. Линейные законы сохранения выражают физические законы, такие как закон сохранения массы или закон баланса заряда. Задача о водород–водородной связи (1.21) и задача Робертсона (1.22) могут рассматриваться как примеры, для которых выполняется этот закон, поскольку при некоторых начальных данных сумма компонент соответствующих решений этих двух задач равна единице. В работе [Shampine, 1998] показано, что все стандартные численные методы решения ЗНУ сохраняют свойство, выражающееся в виде любого из линейных законов сохранения. Например, если сумма компонент решения равна единице, то с точностью до ошибки округления сумма компонент

численной аппроксимации решения также равна единице. Тот факт, что численное решение удовлетворяет одному или нескольким законам сохранения не означает, что эта аппроксимация истинного решения вычислена точно. Действительно, даже для совершенно неадекватных аппроксимаций решения задачи Робертсона, представленных в таблице 1.1, выполнен закон сохранения, заключающийся в том, что сумма компонент этого решения равна единице. Тем не менее, если для численного решения линейный закон сохранения не выполнен с точностью до ошибки округления, то это свидетельствует либо о наличии алгоритмической ошибки в программе, с использованием которой был получен соответствующий результат, либо о том, что решение вычислено с ошибками, обусловленными эффектами, связанными с выполнением вычислений с конечной точностью. Обратимся к рассмотрению свойств решений, которые не сохраняются для их численной аппроксимации, полученной с использованием стандартных методов интегрирования.

В параграфе 1.2 мы установили, что вдоль решений уравнения маятника (1.6) энергия системы сохраняется. Вообще говоря, при численном решении подобных уравнений с использованием стандартных методов, энергия сохраняет свое значение лишь приблизительно. Для того, чтобы убедиться в этом, рассмотрим уравнение маятника, представленное в виде системы уравнений первого порядка. Предположим, что в момент t_n с использованием численной процедуры аппроксимация решения

$$y_{n,1} = \theta(t_n) + e_1, \quad y_{n,2} = \theta'(t_n) + e_2$$

была получена с небольшими ошибками e_1 и e_2. С использованием линеаризованных уравнений можно получить выражение для аппроксимации энергии, соответствующей численному решению

$$\begin{aligned} 0.5(y_{n,2})^2 - \cos(y_{n,1}) &= 0.5(\theta'(t_n) + e_2)^2 - \cos(\theta(t_n) + e_1) \approx \\ &\approx E + \theta'(t_n)e_2 + \sin(\theta(t_n))e_1. \end{aligned}$$

Из этой аппроксимации видно, что ошибка при вычислении энергии сравнима с ошибкой вычисления компонент решения и, следовательно, энергия сохраняет постоянное значение лишь приближенно. Часто этого вполне достаточно, однако в случаях, когда качественные характеристики решения зависят в долгосрочной перспективе от энергии, может оказаться важным обеспечение постоянства ее значения. Одним из возможных путей решения этой задачи является использование стандартных численных процедур, но с заданием повышенной точности вычислений. Этот подход позволяет получить удовлетворительный результат, если моделирование системы выполняется на относительно небольших промежутках времени. Альтернативно можно использовать специальные численные процедуры, основанные на стандартных методах, но в которых решение подвергается некоторым возмущениям, выбранным так, чтобы оно удовлетворяло соответствующим нелинейным законам сохранения. Существуют также численные процедуры, основанные на методах, автоматически обеспечивающих сохранение определенных физических величин, например энергии и момента количества движения. Выбор наиболее эффективного подхода для обеспечения закона сохранения определяется экспериментально. В большинстве случаев выполнение нелинейного закона сохранения вдоль траекторий численных решений может быть обеспечено на практике

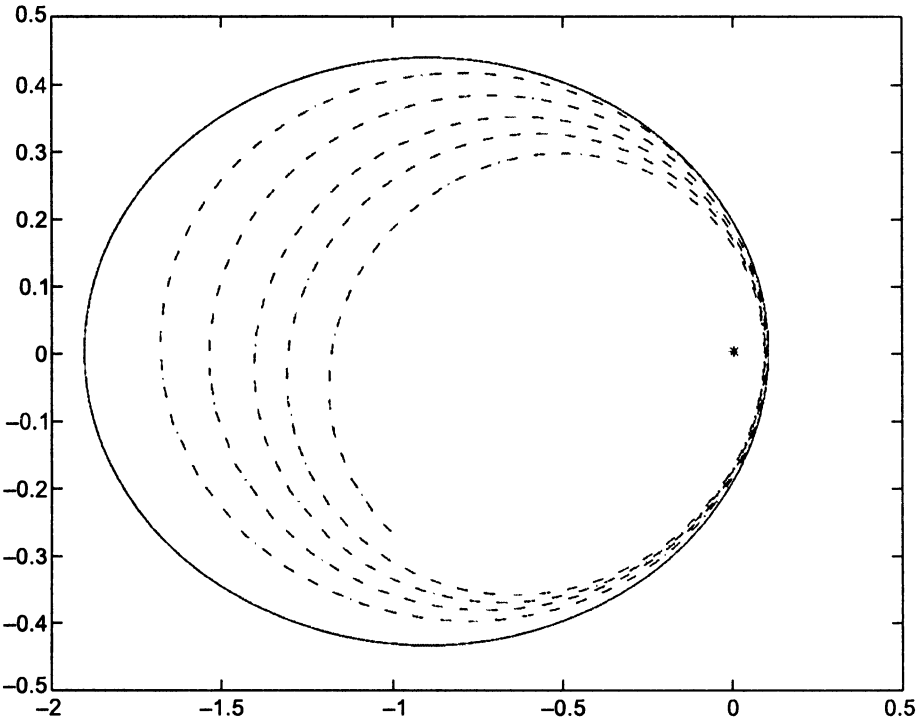

Рис. 1.9. Решение задачи двух тел при $e = 0.9$ на промежутке $0 \leqslant t \leqslant 20$.

либо ценой высоких вычислительных затрат, либо потерей точности вычисления самого решения.

Решение задачи двух тел (1.17) удовлетворяет двум нелинейным законам сохранения: (i) закону сохранения энергии, вычисляемой в соответствии с выражением

$$\frac{x'(t)^2 + y'(t)^2}{2} - \frac{1}{r(t)},$$

где $r(t) = \sqrt{x(t)^2 + y(t)^2}$ — расстояние между телами, и (ii) закону сохранения момента количества движения, определяемого по формуле

$$x(t)y'(t) - y(t)x'(t).$$

На рисунке 1.9 показан график решения системы ОДУ (1.17) с начальными данными (1.18) при эксцентриситете $e = 0.9$. Траектория движения тела, изображенная непрерывной линией, вычислена с заданием большой точности. Показанная пунктиром траектория соответствует численному решению этой задачи при значении допустимой ошибки вычислений, принятой по умолчанию. Неподвижное тело в начале координат показано звездочкой. При решении этой задачи с использованием различных численных процедур и при точности, заданной по умолчанию, значения энергии и момента количества движения вдоль траекторий решений уменьшались соответственно от -0.5000 до -0.7874 и от 0.4359 до 0.3992. В этой физической

задаче постоянное уменьшение энергии соответствует приближению вращающегося тела к неподвижному телу по спиральной траектории. Особенности численного решения зависят только от используемого метода интегрирования, а также от физической сущности задачи, но важно подчеркнуть, что в рассматриваемом случае решение удовлетворяет законам сохранения лишь приближенно. Поэтому на достаточно большом интервале времени интегрирования поведение численного решения может приобрести такие свойства, которые совершенно не соответствуют качественным характеристикам истинного математического решения. В рассматриваемом случае выбранный для интегрирования численный метод допускает постоянное уменьшение энергии и полученный результат можно считать предсказуемым. С другой стороны, если задать большую точность вычислений, найденное численное решение удовлетворяет закону сохранения энергии на промежутке $[0, 20]$. Справедливости ради следует отметить, что небольшое убывание энергии тем не менее имеет место, но соответствующее изменение траектории слишком мало, чтобы быть заметным на экране компьютера.

Как уже отмечалось выше, существуют методы, которые обеспечивают сохранение (по крайней мере приближенно) определенных качественных свойств решения ЗНУ на длительных промежутках интегрирования. Например, для дифференциальных уравнений, определяющих симплектические или обратимые во времени отображения, существует численные методы с соответствующими свойствами. Эти методы позволяют вдоль траекторий численного решения ограничить ошибку при вычислении гамильтониана системы и в некоторых случаях обеспечивают сохранение момента количества движения (см. работы [Sanz-Serna & Calvo, 1994] и [Stuart & Humphries, 1996]). Разумеется, это достигается не бесплатно. При неизменных всех других факторах обеспечение выполнения определенных свойств численного метода, например выполнение свойства его симплектичности, потенциально может привести к потере точности вычисления решения соответствующего ОДУ.

■ УПРАЖНЕНИЕ 1.14

Дифференциальные уравнения

$$\begin{aligned} y_1' &= -y_1, \\ y_k' &= (k-1)y_{k-1} - ky_k, \quad k = 2, 3, \ldots, 9, \\ y_{10}' &= 9y_9 \end{aligned}$$

описывают эволюцию некоторой химической реакции. Покажите, что эта система ОДУ удовлетворяет линейному закону сохранения. В частности, покажите, что сумма компонент решения постоянна.

■ УПРАЖНЕНИЕ 1.15

Модель Волтерра, описывающая эволюцию системы хищник–жертва, может быть записана в виде следующих ОДУ

$$\begin{aligned} x' &= a(x - xy), \\ y' &= -c(x - xy). \end{aligned}$$

Покажите, что решения этой системы удовлетворяют нелинейному закону сохранения

$$G(t,x,y) = x^{-c}y^{-a}e^{cx+ay} = \text{constant}.$$

Напишите программу численного решения этих дифференциальных уравнений с использованием метода Эйлера с постоянным шагом интегрирования h. Найдите численное решение этой ЗНУ на промежутке $0 \leqslant t \leqslant 10$ при значениях параметров $a = 2$, $c = 1$ и начальных значениях $x(0) = 1$, $y(0) = 3$. Постройте фазовый портрет системы, т. е. выведите график траектории $(x(t), y(t))$. Кроме того, выведите график сохраняющейся величины $G(t, x(t), y(t))$. Согласно известным теоретическим результатам величина G постоянна и решение является периодическим, т. е. траектория на фазовой плоскости должна быть замкнутой. Подберите экспериментально значение шага интегрирования h, при котором значение G приблизительно сохраняет свое значение и построенная траектория решения является замкнутой. После изучения численных методов решения ЗНУ, представленных в следующей главе, можно вернуться к рассмотрению этой задачи и решить ее с использованием численной процедуры `ode45`.

Глава 2

ЗАДАЧИ С НАЧАЛЬНЫМИ УСЛОВИЯМИ

§ 2.1. ВВЕДЕНИЕ

В этой главе мы рассмотрим задачу с начальными условиями (ЗНУ) для обыкновенных дифференциальных уравнений (ОДУ). Поскольку ОДУ могут быть записаны по-разному, с теоретической и практической точек зрения представляется целесообразным записать эти уравнения в стандартной форме. В главе 1 были рассмотрены методы, позволяющие записать ОДУ в векторном виде системы уравнений первого порядка

$$y' = f(t, y). \tag{2.1}$$

Далее мы будем предполагать, что рассматриваемые в этой главе ОДУ представлены именно в этой форме. Тем не менее, численные процедуры MATLAB решения ЗНУ допускают представление исследуемого уравнения в виде $M(t, y)y' = f(t, y)$, и этот специальный случай будет рассмотрен в подпараграфе 2.3.2. Мы будем также предполагать, что ОДУ определены на конечном интервале $a \leqslant t \leqslant b$ и соответствующие начальные условия заданы в виде вектора

$$y(a) = A. \tag{2.2}$$

Численные процедуры решения ЗНУ обеспечивают получение приближенного решения, начинающегося в точке $y_0 = A = y(a)$ и проходящего в моменты времени $a = t_0 < t_1 < \ldots < t_N = b$ через последовательные точки его аппроксимации $y_n \approx y(t_n)$. Численные методы могут быть классифицированы по следующему признаку. Если аппроксимация решения в момент t_n определяется с учетом аппроксимаций $y_{n-1}, y_{n-2}, \ldots$, вычисленных на предыдущих шагах, то эти методы называются методами с памятью (многошаговыми). В противном случае, соответствующие методы называются одношаговыми. Задачи с начальными условиями могут быть классифицированы как жесткие и нежесткие. Понятие жесткости ЗНУ трудно определить формально, но соответствующие признаки можно легко выявить. Следует подчеркнуть, что установление принадлежности исследуемой задачи к классу жестких задач имеет важное значение при выборе численного метода ее решения. Система MATLAB предлагает пользователю множество численных методов решения ЗНУ, но общепринято первоначально использовать процедуру `ode45`, основанную на одношаговом явном методе Рунге–Кутта. Если при тестовом решении рассматриваемого уравнения возникло подозрение, что соответствующая задача является жесткой, или если при использовании процедуры `ode45` были

получены неудовлетворительные результаты, рекомендуется применить процедуру ode15, основанную на методах дифференцирования назад (МДН). При обсуждении численных методов решения ЗНУ мы сконцентрируем наше внимание на указанных двух процедурах, поскольку они наиболее широко используются при проведении научных вычислений. Кроме того, для полноты изложения мы также рассмотрим и другие методы, например метод Адамса, реализованный в численной процедуре ode113, и который часто оказывается более предпочтительным по сравнению с явным методом Рунге–Кутта. В заключительном параграфе этой главы представлены практические примеры решения ЗНУ с использованием численных процедур MATLAB. Читатель может изучать теорию численных методов, представленную в первых параграфах этой главы, попутно решая соответствующие задачи из заключительного параграфа.

§ 2.2. ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ ЗНУ

В этом параграфе мы детально рассмотрим численные процедуры решения ЗНУ ode45 и ode15, основанные соответственно на явном методе Рунге–Кутта для нежестких ЗНУ и на методе дифференцирования назад (МДН). Эти процедуры относятся к наиболее эффективным и часто используемым. Мы также вкратце упомянем и другие методы в случаях, когда это может оказаться полезными для дальнейшего изложения. Несмотря на краткость последующего изложения, в нем будут в достаточной мере раскрыты наиболее важные аспекты практического использования упомянутых численных процедур решения ЗНУ.

Численное решение ЗНУ (2.1), (2.2) на интервале $a \leqslant t \leqslant b$ происходит пошагово. Получаемое при этом численное решение начинается в точке $y_0 = A$ и проходит последовательно через точки $y_n \approx y(t_n)$, вычисленные в моменты времени

$$a = t_0 < t_1 < \ldots < t_N = b.$$

Значение y_{n+1} вычисляется в момент времени t_{n+1}, отстоящий от предыдущего момента t_n на величину шага интегрирования $h_n = t_{n+1} - t_n$. Для упрощения записи мы будем использовать для шага интегрирования обозначение $h = h_n$. В точке (t_n, y_n) локальное решение $u(t)$ определяется как решение ОДУ

$$u' = f(t, u),\ u(t_n) = y_n.$$

Из стандартной теории ОДУ следует, что если $v(t)$ и $w(t)$ представляют собой решения (2.1) и если $f(t, y)$ удовлетворяет условию Липшица с константой L, то при $\alpha < \beta$ выполнено неравенство

$$\|v(\beta) - w(\beta)\| \leqslant \|v(\alpha) - w(\alpha)\| e^{L(\beta-\alpha)}.$$

В стандартном случае, когда величина $L(b-a)$ имеет сравнительно небольшое значение, из представленного выше результата следует, что ЗНУ (2.1), (2.2) является устойчивой. Однако приведенное выше неравенство является лишь достаточным условием устойчивости задачи. Например, жесткие задачи являются устойчивыми при $L(b-a) \gg 1$. Без выполнения некоторых дополнительных вычислений очень трудно заранее сказать, что устойчивая ЗНУ является жесткой. При установлении

этого свойства полезно выявить следующие признаки. Жесткая задача является в значительной степени устойчивой в том смысле, что некоторые решения соответствующего ОДУ, начинающиеся вблизи исследуемого решения, сходятся к этому решению очень быстро (здесь понятие «очень быстро» означает, что расстояние между соответствующими точками двух сходящихся решений мало по сравнению с интервалом времени интегрирования $b-a$). Иными словами, эти решения изменяются очень быстро по сравнению с интересующим нас решением, но при этом последнее изменяется сравнительно медленно.

Большинство численных методов позволяют получить аппроксимацию решения только в узловых точках $t_0, t_1, \ldots$, однако соответствующие численные процедуры — включая те, что используются в MATLAB — допускают простую модификацию, в результате которой можно получить аппроксимации решения между узловыми точками. МДН основаны на полиномиальной интерполяции, вследствие чего обеспечивают непосредственное получение кусочно-полиномиальной функции $S(t)$, аппроксимирующей решение $y(t)$ на всем промежутке $[a,b]$. Явный метод Рунге–Кутта не позволяет получить решение в виде полиномиальной интерполяции и в этом смысле МДОБ является более предпочтительным методом. Однако существует модификация метода Рунге–Кутта, с использованием которой на каждом интервале $[t_n, t_{n+1}]$ решение $y(t)$ можно аппроксимировать полиномом. Эту модификацию мы будем называть непрерывным обобщением метода Рунге–Кутта. В более общей постановке вопроса мы будем использовать этот термин в контексте кусочно-полиномиальной аппроксимации решения $S(t)$, определенного на всем интервале $[a,b]$.

2.2.1. Одношаговые методы

В одношаговых методах для вычисления решения в очередной узловой точке используются данные, полученные для текущей узловой точки. В системе MATLAB реализованы несколько численных процедур, основанных на различных классах одношаговых методов, но мы рассмотрим лишь один из этих классов, объединяющий явные методы Рунге–Кутта. При изучении этих явных методов, мы рассмотрим несколько неявных методов, широко используемых при решении ЗГУ. Более подробно эти вопросы будут рассмотрены в главе 3.

В иллюстративных целях полезно рассмотреть следующий частный случай задачи нахождения квадратуры. Рассмотрим ОДУ

$$y' = f(t),\ y(a) = A. \tag{2.3}$$

Для успешного исследования более общих случаев мы должны уметь решать подобные сравнительно простые задачи. Более того, вследствие их простоты, они упрощают понимание существа рассматриваемых далее методов численного решения ОДУ. Локальное решение в узловой точке t_n удовлетворяет уравнению

$$u' = f(t),\ u(t_n) = y_n$$

и, следовательно,

$$u(t_n + h) = y_n + \int_{t_n}^{t_n+h} f(x)dx.$$

Таким образом, вычисление $y_{n+1} \approx u(t_n + h)$ предполагает получение численной аппроксимации значения определенного интеграла.

Основной стратегией численного анализа является использование следующего приема: если поставленную цель исследования невозможно достичь с использованием функции $f(x)$, следует аппроксимировать эту функцию интерполяционным полиномом $P(x)$ и при дальнейшем анализе вместо первоначальной функции использовать эту полиномиальную аппроксимацию. Широко известные формулы аппроксимации определенных интегралов могут быть получены с использованием этого подхода. Например, если аппроксимировать функцию $f(x)$ на интервале $[t_n, t_n + h]$ интерполяционным полиномом с постоянными коэффициентами $P(x) = f(t_n)$, можно приближенно вычислить интеграл от этого полинома

$$\int_{t_n}^{t_n+h} f(x)dx \approx \int_{t_n}^{t_n+h} P(x)dx = hf(t_n),$$

или если интерполяция была выполнена относительно другого конца интервала,

$$\int_{t_n}^{t_n+h} f(x)dx \approx \int_{t_n}^{t_n+h} P(x)dx = hf(t_{n+1}).$$

Аналогично, если используется линейный полином, интерполирующий функцию $f(x)$ в обоих концах интервала

$$P(x) = \left(\frac{(t_n + h) - x}{h}\right) f(t_n) + \left(\frac{x - t_n}{h}\right) f(t_n + h),$$

то при интегрировании мы получаем следующую аппроксимацию

$$\int_{t_n}^{t_n+h} f(x)dx \approx \frac{h}{2}[f(t_n) + f(t_n + h)].$$

С геометрической точки зрения, в первых двух случаях значение интеграла аппроксимируется площадью прямоугольника, а в третьем случае — площадью трапеции (соответствующая формула называется формулой трапеций).

Точность этих аппроксимаций может быть определена с использованием стандартных результатов для полиномиальной аппроксимации гладких функций (см. например [Shampine, Allen, & Pruess, 1997]). Если $P(x)$ — полином степени меньше s, интерполирующий гладкую функцию $f(x)$ в s различных узловых точках $t_{n,j} = t_n + \alpha_j h$, принадлежащих интервалу $[t_n, t_n + h]$,

$$P(t_{n,j}) = f(t_{n,j}),\ j = 1, 2, \ldots, s,$$

то

$$P(x) = \sum_{j=1}^{s} f_{n,j} \prod_{i=1, i\neq j}^{s} \frac{x - t_{n,i}}{t_{n,j} - t_{n,i}}$$

и, для каждой точки x, принадлежащей интервалу $[t_n, t_n + h]$, существует точка ξ, принадлежащая тому же интервалу, такая что

$$f(x) - P(x) = \frac{f^{(s+1)}(\xi)}{(s+1)!} \prod_{j=1}^{s} (x - t_{n,j}).$$

Если функция $f(x)$ достаточно гладкая, присутствующая в этом выражении производная является ограниченной функцией на этом интервале. Следовательно, существует константа C, такая что

$$|f(x) - P(x)| \leqslant Ch^s$$

для всех $x \in [t_n, t_n + h]$. При исследовании изменения поведения решения при различных шагах интегрирования h, мы будем использовать общепринятые обозначения и терминологию, игнорируя значение этой константы. Запишем

$$f(x) - P(x) = O(h^s)$$

или, эквивалентно, $f(x) = P(x) + O(h^s)$. В этом случае говорят, что разность между $f(x)$ и $P(x)$ является «О большое от h в степени s» или, коротко, указанная разность имеет порядок s. С использованием теоремы о точности полиномиальной интерполяции можно легко показать, что

$$\int_{t_n}^{t_n+h} f(x)dx = \int_{t_n}^{t_n+h} P(x)dx + O(h^{s+1}).$$

Применяя этот результат к приведенной выше аппроксимации определенного интеграла площадью прямоугольника, получаем формулу

$$y_{n+1} = y_n + hf(t_n), \tag{2.4}$$

с использованием которой можно получить соответствующее выражение для локальной ошибки

$$u(t_n + h) - y_{n+1} = \int_{t_n}^{t_n+h} f(x)dx - hf(t_n) = O(h^2),$$

порядок которой равен порядку величины

$$y_{n+1} = y_n + hf(t_{n+1}). \tag{2.5}$$

Для аппроксимации по формуле трапеций имеем

$$y_{n+1} = y_n + h\left[\tfrac{1}{2}f(t_n) + \tfrac{1}{2}f(t_n + h)\right] \tag{2.6}$$

и локальная ошибка удовлетворяет $u(t_n + h) - y_{n+1} = O(h^3)$. Таким образом, использование полиномиальной интерполяции с s узловыми точками приводит к формуле интерполяционной квадратуры

$$\int_{t_n}^{t_n+h} f(x)dx = h\sum_{j=1}^{s} A_j f(t_n + \alpha_j h) + O(h^{p+1}),$$

которой соответствует численный метод, определяемый формулой

$$y_{n+1} = y_n + h\sum_{j=1}^{s} A_j f(t_n + \alpha_j h) \tag{2.7}$$

и характеризующийся локальной ошибкой $O(h^{p+1})$. Из общей теории интерполяции следует, что $p \geqslant s$, но при практическом выполнении аппроксимации интегралов важно иметь в виду, что при специальном выборе узловых точек порядок локальной ошибки может удовлетворять $p > s$. Например, при использовании правила средней точки, соответствующего интерполяции функции $f(x)$ полиномом с постоянными коэффициентами в точках $t_n + 0.5h$, величина локальной ошибки имеет порядок $p = 2$, но не $p = 1$, как это должно следовать из общей теории.

Итак, мы получили специальные формулы вида (2.7) для интегрирования интерполяционных полиномов. Как следует выбрать коэффициенты A_j в этой формуле для того, чтобы повысить точность вычислений? Для определения точности вычислений разложим $u(t_n + h)$ и y_{n+1} в ряды Тейлора в окрестности точки t_n и сравним соответствующие члены. Для локального решения ЗНУ имеем

$$u(t_n + h) = u(t_n) + \sum_{k=1}^{p} h^k \frac{u^{(k)}(t_n)}{k!} + O(h^{p+1}) =$$

$$= y_n + \sum_{k=1}^{p} h^k \frac{f^{(k-1)}(t_n)}{k!} + O(h^{p+1}).$$

Разложение в ряд функции в правой части равенства (2.7) является несколько более сложной задачей. Сначала разложим в ряд Тейлора члены суммы

$$f(t_n + \alpha_j h) = \sum_{r=0}^{p-1} (\alpha_j h)^r \frac{f^{(r)}(t_n)}{r!} + O(h^p).$$

Подставив полученный результат в (2.7), получаем

$$y_{n+1} = y_n + h \sum_{j=1}^{s} A_j \left(\sum_{k=1}^{p} (\alpha_j h)^{k-1} \frac{f^{(k-1)}(t_n)}{(k-1)!} \right) + O(h^{p+1}) =$$

$$= y_n + \sum_{k=1}^{p} h^k \left(\sum_{j=1}^{s} A_j \alpha_j^{k-1} \right) \frac{f^{(k-1)}(t_n)}{(k-1)!} + O(h^{p+1}).$$

Сравнивая эти два разложения, можно заметить, что $u(t_n + h) = y_{n+1} + O(h^{p+1})$, если и только если

$$\frac{1}{k} = \sum_{j=1}^{s} A_j \alpha_j^{k-1}, \quad k = 1, 2, \ldots, p. \tag{2.8}$$

Если используется лишь одна узловая точка (т. е. $s = 1$), из первого равенства (2.8) следует, что $A_1 = 1$. Используя этот результат, из второго равенства получаем

$$\frac{1}{2} = A_1 \alpha_1 = \alpha_1.$$

Если $\alpha_1 \neq \frac{1}{2}$, это равенство не выполняется и $u(t_n + h) = y_{n+1} + O(h^2)$. С другой стороны, если $\alpha_1 = \frac{1}{2}$, это равенство справедливо и третье равенство принимает вид

$$\frac{1}{3} = 1 \times \left(\frac{1}{2} \right)^2.$$

Это равенство не выполняется и поэтому $u(t_n+h) = y_{n+1}+O(h^3)$. Представленное равенство имеет вид рассмотренной ранее формулы прямоугольников. В упражнении 2.1 читателю предлагается определить порядок точности вычислений для двух других формул, являющихся вариантом формул для вычисления квадратуры.

Обсуждение сходимости результатов, получаемых с использованием представленных формул, мы начнем с выяснения вопроса о влиянии фактора устойчивости ОДУ. Если функция f удовлетворяет условию Липшица с константой $L = 0$, из представленного ранее общего результата следует, что если $v(t)$ и $w(t)$ представляют собой решения ОДУ $y' = f(t)$ и если $\alpha < \beta$, то

$$\|v(\beta) - w(\beta)\| \leqslant \|v(\alpha) - w(\alpha)\|.$$

Однако легко доказать более строгий результат. Решение $v(t)$ уравнения $y' = f(t)$ имеет вид

$$v(t) = v(\alpha) + \int_{\alpha}^{t} f(x)dx.$$

Вычисляя это выражение при $t = \beta$ и вычитая аналогичное выражение для $w(t)$, получаем равенство

$$v(\beta) - w(\beta) = v(\alpha) - w(\alpha),$$

из которого следует, что локальная ошибка на одном шаге, выполняемом в точке t_n, приводит к выходу траектории численного решения ОДУ на решение, параллельное решению, проходящему через точку y_n.

Пусть истинная ошибка в t_n удовлетворяет

$$e_n = y(t_n) - y_n.$$

При $y_0 = A$ имеем $e_0 = 0$. В главе 1 при исследовании вопроса о распространении ошибки было получено равенство

$$e_{n+1} = y(t_{n+1}) - y_{n+1} = [u(t_{n+1}) - y_{n+1}] + [y(t_{n+1}) - u(t_{n+1})].$$

Первый член в правой части — это локальная ошибка, которая по предположению ограничена величиной Ch_n^{p+1}. С учетом ограниченности решений ОДУ, правая часть которых удовлетворяет условию Липшица, с использованием справедливого по определению равенства $u(t_n) = y_n$ можно заключить, что второй член удовлетворяет неравенству

$$|y(t_{n+1}) - u(t_{n+1})| \leqslant |y(t_n) - y_n|e^{Lh_n}.$$

Принимая во внимание все полученные оценки, окончательно имеем

$$|e_{n+1}| \leqslant Ch_n^{p+1} + |e_n|e^{Lh_n}.$$

Таким образом, ошибка вычислений имеет двоякую природу. Во-первых, она обусловлена наличием локальной ошибки, которая имеет место на каждом шаге и присуща используемому численному методу. Во-вторых, величина ошибки зависит от распространения ошибки на всех предыдущих шагах, обусловленного устойчивостью исследуемой ЗНУ. Совокупный эффект от действия всех этих факторов

можно легко предсказать при решении задачи о вычислении квадратуры, поскольку в этом случае второй фактор отсутствует, а оценка локальной ошибки может быть просто вычислена как сумма соответствующих величин на каждом шаге. При вычислении квадратуры с постоянным шагом $h = (b-a)/N$, получаем равномерную ошибку

$$|y(t_n) - y_n| = |e_n| \leqslant nCh^{p+1} \leqslant (b-a)Ch^p,$$

т.е. $y_n = y(t_n) + O(h^p)$ для всех n. Поэтому в случаях, когда локальная ошибка имеет порядок $O(h^{p+1})$, мы будем говорить, что соответствующая формула имеет порядок p, т.к. порядок аппроксимации $y(t)$ равен именно этой величине. Представленный результат может быть легко обобщен на случай переменного шага. При этом, если H — максимальная величина шага, то истинная (глобальная) ошибка имеет порядок $O(H^p)$.

Оценка локальной ошибки

Локальная ошибка результата y_{n+1}, полученного с использованием формулы порядка p, определяется равенством

$$le_n = u(t_n + h) - y_{n+1}.$$

Если на этом же шаге применить формулу порядка $p+1$ для вычисления результата y^*_{n+1}, получим

$$\begin{aligned} est &= y^*_{n+1} - y_{n+1} \\ &= [u(t_n + h) - y_{n+1}] - [u(t_n + h) - y^*_{n+1}] \\ &= le_n + O(h^{p+2}). \end{aligned} \tag{2.9}$$

Эта величина представляет собой вычислимую оценку локальной ошибки для формулы меньшего порядка, т.к. le_n имеет порядок $O(h^{p+1})$ и, следовательно, соответствующий член в (2.9) доминирует при достаточно малых h. С другой стороны, оценка ошибки для y_{n+1} может быть получена при сравнении с более точным приближенным решением y^*_{n+1}. В общем случае наиболее трудоемкой с вычислительной точки зрения частью процедуры выполнения очередного шага вычислений является нахождение значений функции $f(t_n + \alpha_j h)$ и поэтому для получения на практике оценок локальной ошибки всегда стремятся найти такую пару формул, в каждой из которых используются одни и те же значения этой функции. Для того, чтобы оценка была достоверной, необходимо чтобы результат сравнительно большого порядка y^*_{n+1} был вычислен с наибольшей точностью. Однако почему тогда при выборе стратегии продолжения интегрирования мы отказываемся от использования более точного результата и предпочитаем воспользоваться менее точным результатом y_{n+1}? Ответ заключается в том, что в последнем случае нам неизвестно точно насколько мала локальная ошибка, но мы можем быть уверены, что она будет меньше оценки локальной ошибки. Продолжение интегрирования с использованием более точного результата y^*_{n+1} называется локальной экстраполяцией. Большинство численных процедур MATLAB, основанных на явных методах Рунге–Кутта, используют локальную экстраполяцию (в частности, процедуры `ode23` и `ode45`).

При выполнении численных процедур решения ЗНУ контролируются величины оценки локальной ошибки. Допустимое значение локальной ошибки τ задается пользователем и если оценка этой локальной ошибки становится слишком большой, выполненный шаг интегрирования считается неприемлемым и выполняется другая попытка с меньшим значением шага. В нашем разложении в ряд величины локальной ошибки мы определили порядок ошибки с использованием только первого ненулевого члена ряда Тейлора. Если в этом разложении сохранить следующий член, получаем

$$u(t_n + h) - y_{n+1} = h^{p+1}\phi(t_n) + O(h^{p+2}). \tag{2.10}$$

Из этого равенства становится видно, как следует подстраивать шаг интегрирования. Если в точке t_n выполнить один раз процедуру интегрирования с шагом σh, то соответствующая локальная ошибка будет удовлетворять равенству

$$\begin{aligned}(\sigma h)^{p+1}\phi(t_n) + O((\sigma h)^{p+2}) &= \sigma^{p+1}h^{p+1}\phi(t_n) + O(h^{p+2}) = \\ &= \sigma^{p+1}est + O(h^{p+2}).\end{aligned} \tag{2.11}$$

Наибольшая величина шага интегрирования, при котором обеспечивается допустимая величина ошибки, соответствует значению σ, удовлетворяющему неравенству $|\sigma^{p+1}est| \approx \tau$, т. е.

$$h\left(\frac{\tau}{|est|}\right)^{1/(p+1)}.$$

При программной реализации численной процедуры шаг интегрирования выбирается так, чтобы оценка локальной ошибки не превышала допустимого значения и поэтому подстройка этого шага выполняется до тех пор, пока не будет выполнено это требование. Однако обеспечение выполнения этого условия может оказаться слишком ресурсоемкой процедурой, и тогда она должна прерываться. Прекращение работы процедуры интегрирования может происходить в случае, когда шаг интегрирования становится слишком малым относительно той точности вычислений, которая может быть обеспечена на данном компьютере. Кроме того, прекращение работы процедуры происходит в случае задания слишком большой относительной точности вычислений (см. главу 1). С другой стороны, если текущее значение шага интегрирования обеспечивает вычисление с допустимым значением локальной ошибки, оно может быть увеличено на следующем шаге процедуры интегрирования. Это позволяет увеличить эффективность проведения вычислений, поскольку при сравнительно большом шаге быстрее достигается конец интервала, на котором требуется выполнить интегрирование. Аналогичный подход может быть использован при выборе значения очередного шага интегрирования. При следующем шаге интегрирования σh, выполняемом в точке t_{n+1}, предсказанное значение ошибки будет определяться равенством

$$\begin{aligned}u(t_{n+1} + \sigma h) - y_{n+2} &= (\sigma h)^{p+1}\phi(t_{n+1}) + O(h^{p+2}) = \\ &= \sigma^{p+1}h^{p+1}\phi(t_n) + O(h^{p+2}) = \\ &= \sigma^{p+1}est + O(h^{p+2}).\end{aligned} \tag{2.12}$$

Именно таким образом реализована стратегия выбора шага в процедурах численного интегрирования. В этой связи следует упомянуть важный с практической точки зрения вопрос об ограничении на максимальное и минимальное значение шага, поскольку нельзя пренебрегать эффектами, связанными с изменением значений членов высокого порядка (члены порядка «О большое» в равенствах (2.11) и (2.12)) при достаточно большом изменении величины шага интегрирования. Поскольку при принятии решения о неприемлемости выбранного шага интегрирования приходится платить относительно высокую цену (время вычислений оказывается потраченным впустую), а также с учетом того обстоятельства, что оценка ошибки, используемая для предсказания величины очередного шага интегрирования, может быть неточна, в численных процедурах используется дробная величина от предсказанного значения очередного шага интегрирования. В качестве соответствующего коэффициента обычно используются 0.8 или 0.9. Это делается для того, чтобы при обеспечении требуемой точности вычислений минимизировать шанс возникновения ситуации, когда выбранный шаг интегрирования оказывается неприемлемым. Во времена, когда не была разработана стратегия оценки и управления величиной локальной ошибки, использовался постоянный шаг интегрирования. Этот подход применяется и сегодня, но оценивание и контроль величины локальной ошибки является на практике чрезвычайно важным аспектом численного анализа и используется при решении каждого примера в этой книге, за исключением тех случаев, когда рассматриваются специфические вопросы, связанные с интегрированием с постоянным шагом. Выполнение оценивания и контроль величины локальной ошибки дает исследователю некоторую уверенность в том, что полученная аппроксимация решения ЗНУ является достоверной. Более того, выполнение оценивания локальной ошибки не является затратным с вычислительной точки зрения процессом. В общем случае соответствующие вычислительные затраты окупаются тем, что в результате не приходится ограничивать себя использованием только малых шагов интегрирования для получения заведомо достоверных решений на всем промежутке интегрирования или для обеспечения устойчивости процесса численного решения исследуемого ОДУ. При решении широкого класса ЗНУ, например жестких задач, непрактично использовать постоянный шаг интегрирования. В этой связи полезно вспомнить задачу о протонном переносе, рассмотренную в параграфе 1.4. Ее решение имеет интересную особенность в момент времени приблизительно 10^{-10} (см. рис. 1.7) и для ее выявления шаг интегрирования должен быть выбран сравнимым с этой величиной, но задача рассматривается на интервале длиной 10^6. При использовании постоянного шага интегрирования для решения задачи на этом интервале потребовалось бы выполнить приблизительно 10^{16} шагов, что представляется нереализуемым даже на самых быстрых на сегодняшний день компьютерах. Более того, если бы мы выполнили эти вычисления, полученное решение было бы недостоверным вследствие накопленной на таком большом количестве шагов погрешности формулы. Для получения решения, изображенного на рисунках 1.6 и 1.7, потребовалось всего лишь 10^2 шагов, длина которых изменялась в процессе вычислений от $7 \cdot 10^{-14}$ вблизи указанной особенности до $4 \cdot 10^4$ в области, где решение менялось медленно. Для выполнения этих вычислений на компьютере с процессором Pentium II (433MHz) потребовалось несколько секунд

и при этом эффектов, связанных с наличием погрешности используемой численной формулы, не наблюдалось.

Методы Рунге–Кутта

В этом параграфе мы кратко рассмотрим решение задачи вычисления квадратуры с использованием современных явных методов Рунге–Кутта. Численное решение ЗНУ в общем случае выполняется согласно процедуре, во многом аналогичной той, что будет представлена ниже. Поэтому мы будем концентрировать внимание читателя лишь на отличиях от общей схемы. Локальное решение ЗНУ удовлетворяет уравнению

$$u' = f(t,u),\ u(t_n) = y_n.$$

При интегрировании получаем

$$u(t_n + h) = y_n + \int_{t_n}^{t_n+h} f(x,u(x))dx.$$

В рассматриваемом случае важным отличием от общей схемы является то обстоятельство, что неизвестное локальное решение присутствует в обеих частях представленного выше равенства. При аппроксимации интеграла с использованием квадратурной формулы получаем

$$\int_{t_n}^{t_n+h} f(x,u(x))dx = h\sum_{j=1}^{s} A_j f(t_{n,j}, u(t_{n,j})) + O(h^{p+1}), \tag{2.13}$$

где для простоты записи мы использовали $t_n + \alpha_j h = t_{n,j}$. Эта формула представляется бесполезной, поскольку промежуточные значения $u(t_{n,j})$ не известны. Прежде чем перейти к обсуждению двух основных подходов к решению этой задачи, рассмотрим несколько важных примеров использования численных формул, в которых не используется промежуточное значение.

Аппроксимация интеграла прямоугольником слева имеет вид

$$\int_{t_n}^{t_n+h} f(x,u(x))dx = hf(t_n,u(t_n)) + O(h^2) = hf(t_n,y_n) + O(h^2).$$

Соответствующая численная формула

$$y_{n+1} = y_n + hf(t_n,y_n) \tag{2.14}$$

известна как (прямой, явный) *метод Эйлера*. В контексте задачи решения зависящих от времени дифференциальных уравнений в частных производных эта формула соответствует так называемому *полностью явному методу* (fully explicit method). Аппроксимация

$$y_{n+1} = u(t_n + h) + O(h^2)$$

известна как явная формула Рунге–Кутта первого порядка. Аппроксимация интеграла прямоугольником справа соответствует формуле для *обратного (неявного) метода Эйлера*

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}), \tag{2.15}$$

которая в контексте решения ДУЧП соответствует так называемому *полностью неявному методу*. Реализация этого метода связана с некоторой проблемой, отсутствующей в случае рассмотрения стандартных задач вычисления квадратуры. Существо этой проблемы заключается в том, что новое приближенное решение y_{n+1} определяется неявно в виде решения системы алгебраических уравнений. Точность вычислений по этой формуле соответствует точности, которая может быть достигнута с использованием прямого метода Эйлера и поэтому возникает вопрос: Каковы преимущества использования неявной формулы с учетом возникновения дополнительных вычислительных затрат? Одним из возможных ответов на этот вопрос является то, что использование неявной формулы позволяет решать жесткие задачи более эффективно. В этих случаях обратный метод Эйлера является численным методом наименьшего порядка, принадлежащим к семейству методов дифференцирования назад (МДН), рассмотрению которых посвящен следующий раздел. Члены этого семейства, имеющие порядок k, образуют подсемейство, обозначаемое МДНk. Таким образом, обратный метод Эйлера принадлежит семейству МДН1.

Формула трапеций

$$y_{n+1} = y_n + h\left[\tfrac{1}{2}f(t_n, y_n) + \tfrac{1}{2}f(t_{n+1}, y_{n+1})\right] \tag{2.16}$$

соответствует неявному методу Рунге–Кутта второго порядка, реализованному в MATLAB в виде численной процедуры `ode23t`. В контексте решения ДУЧП эта формула соответствует методу Кранка–Николсона (Crank–Nicolson method). Заметим, что в записи формулы трапеций значения y_n и y_{n+1} равноправны и поэтому эта и подобные ей формулы называются *симметричными*. Направление интегрирования существенно при решении ЗНУ, а при решении ЗГУ обычно не имеет значения. Поскольку симметричные формулы могут быть использованы при любом выборе направления интегрирования, при решении ЗГУ часто применяются именно эти формулы.

Возвращаясь к рассмотрению проблемы, связанной с тем, что промежуточные значения не известны, предположим, что имеется некоторая формула, которая позволяет вычислить аппроксимации $y_{n,j} = u(t_{n,j}) + O(h^p)$. Формулируя это предположение более формально с учетом точности вычислений, предположим, что существует константа C, такая что

$$\|u(t_{n,j}) - y_{n,j}\| \leqslant Ch^p.$$

Принимая во внимание ранее сделанное предположение о том, что функция f в правой части ОДУ удовлетворяет условию Липшица с константой L, из предыдущего неравенства следует

$$\|f(t_{n,j}, u(t_{n,j})) - f(t_{n,j}, y_{n,j})\| \leqslant L\|u(t_{n,j}) - y_{n,j}\| \leqslant LCh^p. \tag{2.17}$$

Если в этой формуле заменить значения функции $f(t_{n,j}, u(t_{n,j}))$ на аппроксимации $f(t_{n,j}, y_{n,j})$, то с учетом (2.13) можно записать

$$\int_{t_n}^{t_n+h} f(x, u(x))dx = h\sum_{j=1}^{s} A_j f(t_{n,j}, y_{n,j}) + O(h^{p+1}).$$

Таким образом, мы получили формулу для численного метода

$$y_{n+1} = y_n + h\sum_{j=1}^{s} A_j f(t_{n,j}, y_{n,j}),$$

из которой видно, что если известны формулы для вычисления промежуточных значений $y_{n,j}$ с точностью до $O(h^p)$, то можно получить формулу для вычисления аппроксимации решения y_{n+1} с точностью до $O(h^{p+1})$.

Существует два основных подхода к нахождению формул для вычисления промежуточных значений. Одним из них является использование формул, аналогичных тем, что используются при вычислении y_{n+1}. Этот подход приводит к неявной формуле Рунге–Кутта и системе алгебраических уравнений, для решения которой в общем случае необходимо вычислить аппроксимацию решения y_{n+1} и промежуточные значения $y_{n,j}$, $j = 1, 2, \ldots, s$. Для успешного применения этой формулы в задаче решения жестких ЗНУ требуется, чтобы она была неявной лишь до определенной степени, но нахождение на одном шаге численной процедуры всех $y_{n,j}$ не является необходимым условием реализуемости этого подхода. Численная процедура решения ЗНУ `ode23tb` основана на неявной формуле, согласно которой на одном шаге процедуры интегрирования вычисляется лишь одно промежуточное значение. Второй подход основан на использовании явных формул. Если для всех $y_{n,j}$ найдены явные формулы, можно получить явную формулу для вычисления y_{n+1}. Явные формулы часто используются при решении нежестких задач. Численные процедуры `ode23` и `ode45` основаны на использовании формул этого типа.

Явные формулы Рунге–Кутта

С использованием явного метода Эйлера можно определить промежуточные значения, которые необходимы для выполнения вычислений по любой квадратурной формуле порядка $p = 2$. В результате этого можно получить формулу, для которой справедливо $u(t_n + h) = y_{n+1} + O(h^3)$. Например, если в качестве квадратурной формулы используется формула трапеций, мы приходим к методу Хойна второго порядка (Heun's method):

$$\begin{aligned} y_{n,1} &= y_n + hf(t_n, y_n), \\ y_{n+1} &= y_n + h\left[\tfrac{1}{2}f(t_n, y_n) + \tfrac{1}{2}f(t_{n+1}, y_{n,1})\right]. \end{aligned} \tag{2.18}$$

В упражнении 2.10 читателю предлагается решить с использованием этой формулы некоторую ЗНУ, а в упражнении 2.2 — найти формулу, позволяющую выполнить эти вычисления с той же точностью, но основанную не на формуле трапеций, а на формуле прямоугольников. Применяя метод Хойна, можно найти промежуточные значения, необходимые для вычислений по любой квадратурной формуле порядка

$p = 3$. Действуя подобным образом, рекурсивно можно получить формулу порядка 4 и формулы более высоких порядков и, следовательно, интерполяционные квадратурные формулы любого порядка. В частности, с использованием этой техники можно найти явную формулу Рунге–Кутта любого порядка.

Если анализ проводить с учетом промежуточных значений, можно заметить, что полученные выше формулы задают явную процедуру, в соответствии с которой из начального равенства

$$y_{n,1} = y_n, \quad f_{n,1} = f(t_n, y_{n,1}) \tag{2.19}$$

получаются равенства для $j = 2, 3, \ldots, s$

$$y_{n,j} = y_n + h_n \sum_{k=1}^{j-1} \beta_{j,k} f_{n,k}, \quad f_{n,j} = f(t_n + \alpha_j h_n, y_{n,j}) \tag{2.20}$$

и, наконец,

$$y_{n+1} = y_n + h_n \sum_{j=1}^{s} \gamma_j f_{n,j}. \tag{2.21}$$

Легко видеть, что равенство (2.19) может быть получено из равенства (2.20) при $j = 1$ и $\alpha_1 = 0$. Тем не менее, мы записали (2.19) отдельной строкой, чтобы подчеркнуть то обстоятельство, что начальными данными для представленной численной процедуры являются значения локального решения $u(t_n) = y_n$ и его производной

$$u'(t_n) = f(t_n, u(t_n)) = f(t_n, y_n)$$

в начальной точке шага численного интегрирования. В качестве меры вычислительной трудоемкости подобной явной формулы удобно использовать число вычислений значения функции $f(x, y)$, которое часто называют числом стадий аппроксимации $f_{n,j}$. В рассматриваемом случае это число равно s.

Порядок численной формулы, определяемой равенствами (2.19), (2.20) и (2.21), может быть определен так же, как это было сделано в отношении формулы (2.7), соответствующей стандартной задаче о квадратуре. Действительно, если при решении этой задачи использовать представленные выше формулы, то во введенных в этом параграфе обозначениях условия (2.8) могут быть представлены в следующем виде

$$\frac{1}{k} = \sum_{j=1}^{s} \gamma_j \alpha_j^{k-1}, \quad k = 1, 2, \ldots, p. \tag{2.22}$$

Условия на коэффициенты формулы Рунге–Кутта, при выполнении которых эта формула имеет порядок p, называются *условными уравнениями*. Равенства (2.22) принадлежат к числу условных уравнений и, следовательно, их выполнение является необходимым, но определенно не достаточным условием. В общем случае анализ усложняется двумя обстоятельствами: во-первых, выражение для производной $f(t, u(t))$ содержит искомое решение $u(t)$, во-вторых $u(t)$ может быть векторной величиной. Для лучшего понимания этих проблем рассмотрим первый нетривиальный член разложения $u(t_n + h)$ в ряд Тейлора в окрестности t_n

$$u(t_n + h) = u(t_n) + u'(t_n)h + u''(t_n)\frac{h^2}{2} + \ldots.$$

С учетом начальных данных для рассматриваемой ЗНУ можно получить значения первых двух членов $u(t_n) = y_n$ и $u'(t_n) = f(t_n, u(t_n)) = f(t_n, y_n)$. Вычисление следующего члена является более трудной задачей. Если рассматриваемая система состоит из d уравнений, i-я компонента локального решения удовлетворяет равенству

$$u_i'(t) = f_i(t, u_1(t), u_2(t), \ldots, u_d(t))$$

и, следовательно,

$$u_i''(t) = \frac{\partial f_i}{\partial t} + \sum_{j=1}^{d} \frac{\partial f_i}{\partial u_j}\frac{du_j}{dt} = \frac{\partial f_i}{\partial t} + \sum_{j=1}^{d} \frac{\partial f_i}{\partial u_j} f_j.$$

Для определения последующих членов необходимо сделать предположения, которые позволили бы упростить анализ. В частности, эта цель может быть достигнута, если предположить, что функция $f(t, y)$ не зависит от t. Подобные уравнения называются *автономными*. Поскольку любая ЗНУ может быть сведена к автономной ЗНУ, это предположение с теоретической точки зрения не представляется ограничительным. Несколько отклоняясь от основной линии изложения, мы представим ниже метод, позволяющий свести ЗНУ общего вида к автономной ЗНУ. Рассмотрим ОДУ

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

на интервале $a \leqslant t \leqslant b$. Введем новую независимую переменную $x = t$ (т. е. сделаем переменную t зависимой). Тогда ОДУ

$$\frac{d\mathrm{Y}}{dx} = \frac{d}{dx}\begin{pmatrix} y \\ t \end{pmatrix} = \begin{pmatrix} f(t, y) \\ 1 \end{pmatrix} = F(Y), \; Y(a) = \begin{pmatrix} A \\ a \end{pmatrix}$$

определено на $a \leqslant x \leqslant b$. Этот подход вполне применим, поскольку большинство стандартных методов позволяют интегрировать уравнение для переменной t достаточно точно. Возвращаясь к разработке метода интегрирования, можно заметить, что нам необходимо дополнительно ввести некоторые новые обозначения, связанные с рекурсивным вычислением производных. Для целей этой книги не требуется детальное изложение этих вопросов и поэтому мы рассмотрим лишь специальный случай квадратурной формулы. В упражнении 2.3 читателю предлагается детально исследовать другой частный случай — явную формулу Рунге–Кутта второго порядка.

Выше была представлена методология, позволяющая получить явные формулы Рунге–Кутта любого порядка, но построенные таким образом формулы, вообще говоря, не являются эффективными. Значительные усилия исследователей были направлены в свое время на нахождение формул, которые имели бы заданный порядок p при минимальном количестве стадий аппроксимации. В результате было установлено, что для формул порядка $p = 1, 2, 3$ и 4, минимальным числом стадий аппроксимации является $s = p$; для формул порядка $p = 5$ эта величина равна $s = 6$. Минимизация числа стадий аппроксимации является интересной задачей, но ее важность не настолько велика, как может показаться на первый взгляд. Дополнительные стадии аппроксимации могут быть введены в алгоритм

Таблица 2.1: Таблица Бутчера для формул Рунге–Кутта

α	β
	γ

Таблица 2.2: Таблица Бутчера для $(1,2)$-пары формул Эйлера–Хойна

0		
1	1	
	1	
	$\frac{1}{2}$	$\frac{1}{2}$

для увеличения точности вычислений, обеспечиваемой соответствующей формулой. При выполнении вычислений с использованием более точной (но и более трудоемкой с вычислительной точки зрения) формулы, появляется возможность использовать сравнительно большой шаг интегрирования. В этом случае эффект от уменьшения общего числа необходимых вычислений функции $f(t,y)$ может превзойти эффект от увеличения вычислительных затрат на отдельном шаге. Кроме этого основного обстоятельства необходимо также учитывать не столько трудоемкость вычислений по формуле рассматриваемого метода вычислений, сколько трудоемкость вычислений пары формул, определяющих один шаг интегрирования, а также трудоемкость вычисления оценки локальной ошибки. Проблемы, связанные с вычислением оценки локальной ошибки (2.9), имеют место не только в задачах вычисления квадратуры. Нахождение пары формул, определяющих один шаг интегрирования и использующих при этом одни и те же значения, полученные на стадиях аппроксимации, представляет собой важную задачу. Выше было отмечено, что для выполнения вычислений по формуле порядка $p = 5$ необходимо не менее шести стадий аппроксимации $s = 6$. Известно, что при выполнении вычислений пары формул порядка 4 и 5 — $(4,5)$-пары — используется в совокупности шесть стадий аппроксимации. Например, в часто используемой паре формул, предложенной Фехлбергом [Fehlberg, 1970] и обозначаемой F(4,5), используется шесть стадий аппроксимации. В $(4,5)$-паре, предложенной Дорманом и Принсом [Dormand and Prince, 1980], обозначаемой DOPRI5 и используемой в численной процедуре `ode45`, имеется семь стадий аппроксимации. Дополнительное вычисление функции используется для повышения точности вычислений и тесты действительно показывают, что DOPRI5 несколько превосходит по этой характеристике F(4,5).

Коэффициенты, определяющие формулу Рунге–Кутта, представлены в таблице 2.1 (так называемая таблица Бутчера). В случае явного метода Рунге–Кутта все диагональные и наддиагональные элементы матрицы β равны нулю и на соответствующих местах таблицы принято ничего не писать. Если таблица Бутчера выписывается для пары формул, векторы коэффициентов γ принято писать один

Таблица 2.3: Таблица Бутчера для BS(2,3)-пары

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{3}{4}$	0	$\frac{3}{4}$		
1	$\frac{2}{9}$	$\frac{1}{3}$	$\frac{4}{9}$	
	$\frac{7}{24}$	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{8}$

под другим. Таблица 2.2 представляет собой таблицу Бутчера для ранее рассмотренного простейшего примера подобной пары с минимальным количеством стадий аппроксимации — (1,2)-пары для формул Эйлера и Хойна.

Таблица Бутчера для BS(2,3)-пары формул [Bogacki–Shampine, 1989], реализованной в процедуре `ode23`, представлена в таблице 2.3. Структура этой таблицы несколько отличается от структуры рассмотренных ранее таблиц Бутчера и это обусловлено тем, что BS(2,3)-пара является примером использования подхода «First Same As Last» (FSAL). Для формул, полученных в соответствии с FSAL, значение функции, используемое на первой стадии аппроксимации последующего шага интегрирования, является тем значением функции, которое было получено на последней стадии аппроксимации текущего шага интегрирования. В последней строке таблицы приведены коэффициенты γ в формуле второго порядка с двумя стадиями. В строке, расположенной выше, приведены коэффициенты в формуле для вычисления последней стадии аппроксимации. Эта формула по построению является формулой третьего порядка с тремя стадиями аппроксимации. Таким образом, рассматриваемый метод заключается в выполнении следующих операций: получение результата третьего порядка y_{n+1} с использованием трех стадий аппроксимации, выполнение вычисления для четвертой стадии $f(t_{n+1}, y_{n+1})$ и, наконец, получение результата второго порядка. Эта пара формул была получена с учетом того, что она будет использоваться в методах с локальной экстраполяцией. Таким образом, выполнение очередного шага процедуры интегрирования осуществляется с использованием результата третьего порядка y_{n+1} в качестве приближенного решения. Значение функции, полученное для выполнения вычислений по формуле второго порядка и нахождения оценки локальной ошибки, используется на следующем шаге интегрирования в качестве значения для первой стадии аппроксимации. В этом случае, если результат выполнения шага оказывается приемлемым, общее количество необходимых на каждом шаге стадий аппроксимации уменьшается на единицу. На практике большинство шагов оказываются приемлемыми и вычислительные затраты для этой пары формул несколько больше, чем при использовании подхода, основанного на применении пары формул с тремя стадиями аппроксимации. Очевидно, что нахождение формул, удовлетворяющих свойству FSAL, является непростой задачей, но некоторые широко используемые пары формул принадлежат к этому классу. Примером подобной пары является DOPRI5-пара с семью стадиями, которая реализована в MATLAB в виде численной процедуры `ode45`.

Связанные с использованием этой пары вычислительные затраты на практике оказываются несколько больше, чем соответствующие затраты при вычислении минимум шести значений функции, присутствующих в любой формуле пятого порядка. В широко используемом пакете программ RKSUITE [Brankin, Gladwell & Shampine, 1993] для языка Fortran 77 и в аналогичном пакете `rksuit_90` [Brankin & Gladwell, 1994] для языка Fortran 90 реализован метод, основанный на удовлетворяющей свойству FSAL (4,5)-паре формул с дополнительной стадией аппроксимации. Эта пара формул рассматривается в упражнении 2.4.

Непрерывные обобщения численных методов

Выше были рассмотрены вычислительные методы, которые позволяют вычислить аппроксимации $y(t)$ в точках $t_0 < t_1 < \ldots$, однако в некоторых случаях необходимо получить аппроксимации решения для всех t. Например, в процедурах для построения графиков соответствующая кривая строится путем соединения последовательно полученных точек решения прямыми линиями, и поэтому для гладкости этой кривой необходимо, чтобы эти точки были расположены достаточно близко друг к другу. Однако в формулах Рунге–Кутта высокого порядка могут использоваться настолько большие шаги интегрирования, что сегменты прямых линий, соединяющие соответствующие точки траектории решения, образуют ломаную линию неприглядного вида. На рисунках 4.28 и 4.29 в работе [Bender & Orszag, 1999], показаны примеры подобных графиков. Аналогичный случай рассматривается в упражнении 2.21. Для получения гладких графиков нам необходимо получить вычислительно малозатратную процедуру аппроксимации решения в промежуточных точках. В контексте применения методов Рунге–Кутта это часто связывают с задачей получения «плотного вывода» (dense output).

Непрерывное обобщение методов Рунге–Кутта — это относительно новый результат. Идея этого подхода заключается в том, что значения функции в правой части уравнения, вычисленные на стадиях аппроксимации (включая дополнительные стадии) для нахождения численного решения и оценки локальной ошибки, используются после выполнения соответствующего шага интегрирования от t_n до $t_n + h$ при нахождении полиномиальной аппроксимации для $u(t)$ на промежутке $[t_n, t_n + h]$. Простейшим случаем применения этого подхода является реализация процедуры `ode23`. В начальной точке каждого шага интегрирования мы имеем аппроксимации значения решения y_n и его производной $y'_n = f(t_n, y_n)$. В конечной точке шага вычисляется аппроксимация y_{n+1}. При использовании BS(2,3)-пары также может быть получена аппроксимация производной $y'_{n+1} = f(t_{n+1}, y_{n+1})$, поскольку эта пара принадлежит классу FSAL. (Это значение всегда вычисляется на каждом шаге применения любой явной формулы Рунге–Кутта, т. к. эта производная совпадает со значением, полученным при вычислении первой стадии аппроксимации на следующем шаге.) Используя значения решения и значения его производных на обоих концах интервала, можно найти соответствующие кубические эрмитовы интерполяции на интервале текущего шага интегрирования. С использованием теории интерполяции можно показать, что полученный кубический интерполяционный полином аппроксимирует локальное решение во всех точках интервала $[t_n, t_{n+1}]$ с той же точностью, с какой y_{n+1} аппроксимирует его в момент

времени t_{n+1}. Значения и производные кубического эрмитова интерполяционного полинома на интервале $[t_{n-1}, t_n]$, вычисленные в конечной точке этого интервала, равны соответствующим значениям и производным в точке t_n для аналогичного интерполяционного полинома, построенного на следующем промежутке $[t_n, t_{n+1}]$. Поэтому все эти полиномы образуют единую кусочно-кубическую аппроксимацию на всем интервале интегрирования $S(t) \in C^1[a,b]$. Этот подход лежит в основе численной процедуры `dde23`, которая будет рассмотрена в главе 4 в контексте решения дифференциальных уравнений с запаздыванием.

Использование интерполяции для непрерывного обобщения формул Рунге–Кутта представляет собой важный результат, но его использование связано с выполнением некоторых требований. Для каждой константы $\sigma \in [0,1]$ интерполяционный полином, вычисленный в точке $t_n + \sigma h$, может рассматриваться как формула Рунге–Кутта для выполнения шага интегрирования длиной σh из точки t_n. Поэтому альтернативным подходом для решения задачи является непосредственное нахождение подобного семейства формул. При этом с точки зрения экономии вычислительных затрат необходимо, чтобы в искомой формуле, предназначенной для выполнения шага длиной σh, использовалось как можно больше значений функции, одновременно используемых в основной формуле метода при выполнении вычислений на шаге интегрирования до точки $t_n + h$. Очевидно, что искомая формула зависит от σ, но указанные значения функции не зависят от этой величины. Кроме того, все дополнительные стадии аппроксимации, введение которых обусловлено необходимостью обеспечения заданного порядка точности формулы, также не должны зависеть от σ. Оказывается, что выполнение всех этих требований возможно и коэффициенты в соответствующих формулах представляют собой полиномы от σ. В частности, подобным образом было получено непрерывное обобщение для процедуры `ode45`. При этом для получения интерполяционных полиномов четвертого порядка не требуется вычислять дополнительные стадии аппроксимации, но кусочно-полиномиальная функция $S(t)$ является непрерывной лишь на интервале $[a,b]$. Формула DOPRI5, представляющая собой основу для вычислительной процедуры `ode45`, позволяет использовать достаточно большие шаги численного интегрирования и на каждом таком шаге интерполяционный полином вычисляется по умолчанию в четырех равноразнесенных промежуточных точках. Полученные значения выводятся наряду с аппроксимаций решения, вычисленной непосредственно по этой формуле. В большинстве ситуаций этих дополнительных значений вполне достаточно для получения гладкого графика решения, но в реализации этой численной процедуры предусмотрена возможность увеличения количества промежуточных точек.

■ УПРАЖНЕНИЕ 2.1

Используя условные уравнения (2.8) для квадратурных задач, проверьте, что следующие методы имеют порядок 4.

- Метод Симпсона основан на использовании квадратурной формулы, известной как трехточечная формула Лобатто (three-point Lobatto formula)

$$\int_a^b f(x)dx \simeq \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right].$$

- Двухточечный метод вычисления квадратуры Гаусса основан на использовании формулы

$$\int_a^b f(x)dx \simeq \frac{b-a}{2}\left[f\left(\frac{a+b}{2}-\frac{b-a}{2\sqrt{3}}\right)+f\left(\frac{a+b}{2}+\frac{b-a}{2\sqrt{3}}\right)\right].$$

■ УПРАЖНЕНИЕ 2.2

Используйте метод Эйлера и формулу прямоугольников для средней точки для вывода явной формулы Рунге–Кутта второго порядка с двумя стадиями аппроксимации.

■ УПРАЖНЕНИЕ 2.3

Для лучшего понимания условных уравнений найдите три таких уравнения, при выполнении которых формула

$$y_{n+1} = y_n + h[\gamma_1 f_{n,1} + \gamma_2 f_{n,2}],$$

где

$$f_{n,1} = f(t_n, y_n),$$
$$f_{n,2} = f(t_n + \alpha_1 h, y_n + h\beta_{1,0} f_{n,1}),$$

имеет второй порядок. На шаге, начинающемся в точке (t_n, y_n), результатом y_{n+1} применения этой формулы является аппроксимация решения уравнения

$$u' = f(t,u),\ u(t_n) = y_n$$

в точке $t_{n+1} = t_n + h$. Разложите $u(t_{n+1})$ и y_{n+1} в окрестности t_n в ряды по степеням h и для получения условных уравнений приравняйте соответствующие члены. Для упрощения разложения в ряды рассмотрите частный случай, когда функция $f(t,u)$ является скалярной функцией.

■ УПРАЖНЕНИЕ 2.4

Явные формулы Рунге–Кутта

$$y_{n+1} = y_n + hf_{n,2},$$
$$y^*_{n+1} = y_n + \frac{h}{9}[2f_{n,1} + 3f_{n,2} + 4f_{n,3}]$$

с тремя стадиями аппроксимации

$$f_{n,1} = f(t_n, y_n),$$
$$f_{n,2} = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}f_{n,1}\right),$$
$$f_{n,3} = f\left(t_n + \frac{3}{4}h, y_n + \frac{3}{4}hf_{n,2}\right)$$

имеют порядок 2 и 3 соответственно. Постройте для этой $(2,3)$-пары соответствующую таблицу Бутчера. Оценка локальной ошибки для формулы меньшего порядка

определяется равенством $est = y^*_{n+1} - y_{n+1}$. Предположим, что интегрирование продолжается с использованием результата большего порядка (локальной экстраполяции решения) и что заданы допустимые значения относительной ошибки τ_r и абсолютной ошибки τ_a. Это означает, что если выполнено неравенство

$$|est| \leqslant \tau_r |y^*_{n+1}| + \tau_a,$$

то точность полученного на текущем шаге решения считается приемлемой. Какой следует использовать шаг $h_{\text{new}} = \sigma h$ в случае, если необходимо заново выполнить вычисления в той же точке вследствие получения слишком большой локальной ошибки? Какой шаг должен использоваться на следующем шаге в случае, если оценка величины локальной ошибки оказалась приемлемой? В некоторых численных процедурах величина ошибки вычисляется в соответствии с приведенной выше формулой, в которой $|y^*_{n+1}|$ заменено на $0.5(|y_n| + |y^*_{n+1}|)$. Чем обусловлена целесообразность этой замены?

2.2.2. Многошаговые методы

Методы Адамса

При численном интегрировании дифференциального уравнения в момент времени t_n в нашем распоряжении имеются вычисленные на предыдущих шагах значения решения $y_n, y_{n-1}, \ldots$ и значения соответствующих производных $f_n = f(t_n, y_n), f_{n-1} = f(t_{n-1}, y_{n-1}), \ldots$ Эти данные используются при вычислении y_{n+1} на основе квадратурных формул, рассмотренных в предыдущем параграфе. Напомним, что в качестве аппроксимации локального решения уравнения

$$u' = f(t, u), \; u(t_n) = y_n$$

используется

$$u(t_n + h) = y_n + \int_{t_n}^{t_n+h} f(x, u(x))dx.$$

Построив по s значениям $f_{n,j} = f(t_{n,j}, y_{n,j})$ интерполяционный полином $P(x)$ и выполнив интегрирование, получаем следующую формулу

$$y_{n+1} = y_n + h \sum_{j=1}^{s} A_j f(t_{n,j}, y_{n,j}).$$

Можно показать, что если значения $y_{n,j} \approx u(t_{n,j})$ вычислены достаточно точно, то эта формула имеет порядок точности по крайней мере s. Для одношаговых методов требуется, чтобы $t_{n,j} \in [t_n, t_n + h]$. Таким образом, основная цель состоит в том, чтобы достаточно точно вычислить $y_{n,j}$. Альтернативно можно положить $t_{n,j} = t_{n-j}$, поскольку к этому моменту обычно уже имеются достаточно точные аппроксимации $y_{n,j} = y_{n-j}$, а также $f_{n-j} = f(t_{n,j}, y_{n,j})$. При использовании этого подхода мы получаем семейство явных формул, называемых *формулами Адамса–Башфорта*. Формула Эйлера обладает наименьшим порядком и принадлежит этому семейству, поскольку она представляет собой результат интерполяции для единственной точки f_n. В принятых обозначениях этой формуле соответствует

аббревиатура AB1. При интерполяции по двум точкам f_n и f_{n-1} мы получаем формулу второго порядка AB2

$$y_{n+1} = y_n + h_n \left[\left(1 + \frac{r}{2}\right) f_n - \left(\frac{r}{2}\right) f_{n-1} \right],$$

где $r = h_n/h_{n-1}$ (см. упражнение 2.5). Заметим, что поскольку в этих многошаговых методах используются значения, вычисленные на предыдущих шагах, интервалы времени между соответствующими точками (шаги интегрирования) явно входят в коэффициенты формул. Поэтому в общем случае на каждом шаге интегрирования необходимо пересчитывать эти коэффициенты. Для повышения эффективности этих вычислений были разработаны специальные методики. При теоретическом анализе этих формул часто предполагают, что шаг интегрирования h является постоянным. В этом случае $r = 1$ и формула AB2 упрощается

$$y_{n+1} = y_n + h \left[\frac{3}{2} f_n - \frac{1}{2} f_{n-1} \right].$$

Формулы Адамса–Моултона могут быть получены аналогичным образом, но интерполяционный полином строится для f_{n+1}. Поскольку равенство $f_{n+1} = f(t_{n+1}, y_{n+1})$ зависит от y_{n+1}, эти формулы являются неявными. Обратный метод Эйлера определяется формулой первого порядка; формула трапеций имеет порядок 2. В принятых обозначениях этим формулам соответствуют аббревиатуры AM1 и AM2.

Точность методов Адамса может быть исследована аналогично тому, как это было сделано ранее для методов Рунге–Кутта. Неявная формула Адамса–Моултона порядка k (AMk) является более точной и более устойчивой, чем соответствующая явная формула Адамса–Башфорта порядка k (ABk). Выбор метода в конкретной ситуации зависит от сложности соответствующего нелинейного алгебраического уравнения, которое приходится решать при использовании неявной формулы. В случае нежестких задач неявные уравнения решаются *методом простых итераций*.

Использование метода простых итераций мы продемонстрируем на примере AM1. Решим итеративно алгебраические уравнения

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}) \tag{2.23}$$

относительно y_{n+1}. Итеративная процедура определяется следующим равенством

$$y_{n+1}^{[m+1]} = y_n + hf(t_{n+1}, y_{n+1}^{[m]}),$$

где $y_{n+1}^{[m]}$ — аппроксимация решения, полученная на шаге m. Задаваемая этой формулой операция является «коррекцией» текущего значения, а сама формула называется *корректором*. В предположении, что уравнение (2.23) имеет решение, из условия Липшица для f следует, что

$$\|y_{n+1} - y_{n+1}^{[m+1]}\| = \|hf(t_{n+1}, y_{n+1}) - hf(t_{n+1}, y_{n+1}^{[m]})\| \leqslant hL\|y_{n+1} - y_{n+1}^{[m]}\|.$$

Из этого равенства следует, что если $hL < 1$, то очередная итерация ближе к искомому решению y_{n+1}, чем предыдущая и, следовательно, рассматриваемый итеративный процесс сходится. Этот результат может быть использован для доказательства того факта, что при достаточно малых шагах интегрирования h алгебраические уравнения (2.23) имеют решение и это решение является единственным. (Метод простых итераций может служить в качестве примера использования известных в прикладной математике теоретических результатов о неподвижных точках отображений.) Чем меньше значение h, тем быстрее сходится этот итеративный процесс. Это обстоятельство представляется очень важным, поскольку на каждом шаге итераций вычисляется функция f в правой части ОДУ и, если окажется, что необходимо сделать слишком большое количество итераций, для решения задачи с той же точностью возможно следует использовать явный метод с меньшим шагом интегрирования. С другой стороны, с точки зрения эффективности вычислений хотелось бы использовать максимально возможный шаг интегрирования. Для уменьшения необходимого числа итераций очень важно задать наиболее близкое к искомому решению y_{n+1} начальное значение $y_{n+1}^{[0]}$ итеративной процедуры. Наиболее точно предсказать это значение можно с использованием явной формулы, которую называют *предиктором*. Для неявной формулы Адамса–Моултона AMk в качестве такого предиктора естественно использовать явную формулу Адамса–Башфорта — либо ABk, либо AB$(k-1)$. Другой важный способ уменьшения вычислительной трудоемкости обусловлен тем фактом, что на практике нет необходимости точно вычислять решение нелинейного уравнения, соответствующего неявной формуле, т. к. точность определения этого решения должна быть достаточной для того, чтобы была обеспечена заданная точность решения основной задачи — задачи численного интегрирования ОДУ. При качественной программной реализации современные алгоритмы численного интегрирования, например процедура VODE [Brown, Byrne & Hindmarsh, 1989], основанная на методах Адамса–Моултона для нежестких задач, вычисляют функцию f всего два раза на каждом шаге. С практической точки зрения метод простых итераций наиболее предпочтителен при решении нежестких задач, поскольку в классической ситуации, когда значение $L(b-a)$ невелико, требование $Lh < 1$ не ограничивает величину шага интегрирования. Для обеспечения заданной точности вычисления решения часто выбирают достаточно малый шаг интегрирования, однако при этом следует учитывать, что это приводит к значительному уменьшению скорости сходимости метода простых итераций.

Существует важный класс модифицированных методов Адамса, называемых *методами предиктор-корректор*. Предсказание очередной аппроксимации делается с использованием формулы Адамса–Башфорта, после чего делается фиксированное количество коррекций с использованием формулы Адамса–Моултона. В наиболее широко используемом методе этого класса коррекция выполняется только один раз, и этот метод основан на формуле Хойна (2.18). Значение $y_{n+1}^{[0]}$ предсказывается с использованием метода Эйлера AB1. В обозначениях, используемых для записи метода Хойна, в качестве этого значения выступала величина $y_{n,1}$. Остальные компоненты метода Хойна можно рассматривать в качестве одной коррекции по формуле трапеций (AM2) для получения решения y_{n+1} и значения $f(t_{n+1}, y_{n+1})$, используемого на следующем шаге. Эта процедура

составляет существо PECE-метода (Predict—Evaluate f—Correct—Evaluate f, т.е. Предсказание—Вычисление f—Коррекция—Вычисление f). Было бы естественно в качестве предиктора для AM2 использовать формулу AB2, однако применение в этом качестве формулы меньшего порядка более предпочтительно. Нетрудно показать, что составленная из формулы предиктора порядка $k-1$ и формулы корректора порядка k формула предиктор–корректор имеет порядок k. Это можно доказать аналогично тому, как это было сделано ранее для квадратурной формулы и, в частности, для метода Хойна. Формула предиктор–корректор имеет тот же порядок, что и используемая в корректоре неявная формула, однако главные члены в соответствующих разложениях ошибки в ряд различны. В упражнении 2.7 рассматривается пример, иллюстрирующий этот факт. Важно понимать, что метод предиктор–корректор принадлежит к классу *явных* методов и обладает качественными свойствами, которые в некотором смысле не соотносятся с неявной формулой, определяющей корректор. Одну из этих особенностей мы рассмотрим, когда коснемся вопроса устойчивости метода предиктор–корректор. Совместное использование формул разного вида часто вызывает трудности для понимания, особенно в случаях рассмотрения нежестких задач, когда неявные формулы вычисляются в процессе предсказания и коррекции решения. На практике это сводится к выбору: либо вы вычисляете неявную формулу столько раз, сколько это необходимо для обеспечения заданной точности решения, либо вы используете фиксированное число итераций. При использовании неявных методов всегда непросто выработать критерий того, что итеративный процесс для нахождения решения нелинейного уравнения, соответствующего неявной формуле, сошелся и при этом была обеспечена достаточная точность. Этой проблемы не существует при использовании явных методов предиктор–корректор. В зависимости от конкретной ситуации результаты применения рассматриваемых здесь двух типов численных методов могут существенно различаться, однако сравнение программных реализаций метода Адамса предиктор–корректор `ODE/STEP`, `INTRP` [Shampine & Gordon, 1975] или `ode113` с программными реализациями неявного метода Адамса–Моултона `DIFSUB` [Gear, 1971] или `VODE` [Brown et al., 1989]) показывает, что первый алгоритм ведет себя так же как и второй. Практическое применение подобных формул (см., например, упражнение 2.10) позволяет выявить различия между методами предиктор–корректор и неявными методами.

Методы дифференцирования назад

В процессе численного интегрирования при достижении момента t_n аппроксимация решения $y(t)$ интерполяционным полиномом $P(t)$, построенным по точке y_{n+1} и вычисленным на предыдущих шагах аппроксимациям $y_n, y_{n-1}, \ldots y_{n+1-k}$, может быть выполнена с использованием *формулы дифференцирования назад*, имеющей порядок точности k (МДНk). Эта полиномиальная аппроксимация должна удовлетворять в момент t_{n+1} рассматриваемому ОДУ (в терминологии, принятой в теории ЗГУ, должна иметь место коллокация аппроксимации с этим ОДУ). Это требование может быть представлено в виде алгебраического уравнения относительно y_{n+1}

$$P'(t_{n+1}) = f(t_{n+1}, P(t_{n+1})) = f(t_{n+1}, y_{n+1}),$$

которое называется уравнением коллокации. В методе МДН1 используется линейная интерполяция в t_{n+1} и t_n. В этом случае производная от интерполяционного полинома равна константе и, следовательно, уравнение коллокации упрощается

$$\frac{y_{n+1} - y_n}{h_n} = f(t_{n+1}, y_{n+1}),$$

или эквивалентно

$$y_{n+1} - y_n = h_n f(t_{n+1}, y_{n+1}).$$

Интерполяция квадратичным полиномом по трем точкам t_{n+1}, t_n и t_{n-1} приводит к методу МДН2 и соответствующему уравнению коллокации

$$\left(\frac{1+2r}{1+r}\right) y_{n+1} - (1+r)y_n + \left(\frac{r^2}{1+r}\right) y_{n-1} = h_n f(t_{n+1}, y_{n+1}),$$

где $r = h_n/h_{n-1}$ (см. упражнение 2.5). Для краткости изложения далее будут рассматриваться методы МДН с шагом h, сохраняющим свое значение на протяжении нескольких шагов процедуры численного интегрирования. В этом случае уравнение коллокации принимает следующий вид

$$\tfrac{3}{2} y_{n+1} - 2y_n + \tfrac{1}{2} y_{n-1} = h f(t_{n+1}, y_{n+1}).$$

В случае, когда шаг интегрирования h равен константе, формулы Адамса и методы МДН принадлежат классу *линейных многошаговых методов* (ЛММ). Определяющие эти методы формулы имеют следующий вид

$$\sum_{i=0}^{k} \alpha_i y_{n+1-i} = h \sum_{i=0}^{k} \beta_i f(t_{n+1-i}, y_{n+1-i}). \tag{2.24}$$

При доказательстве сходимости одношаговых методов было показано, что распространение ошибки вычислений, произошедшей на отдельном шаге, может быть ограничено (в смысле устойчивости ЗНУ), если максимальное значение шага интегрирования стремится к нулю. Доказательство сходимости многошаговых методов представляет собой более сложную задачу, т. к. в этом случае произошедшая на текущем шаге ошибка в значительной степени зависит от ошибок, произошедших на предыдущих шагах, и поэтому приходится проводить анализ аппроксимации не локального, а глобального решения и вместо вопроса об устойчивости ЗНУ рассматривать вопрос об устойчивости самого численного метода. *Ошибка дискретизации* (*локальная погрешность формулы*) (lte_n) линейного многошагового метода определяется равенством

$$lte_n = \sum_{i=0}^{k} \alpha_i y(t_{n+1-i}) - h \sum_{i=0}^{k} \beta_i f(t_{n+1-i}, y(t_{n+1-i})) =$$
$$= \sum_{i=0}^{k} \alpha_i y(t_{n+1-i}) - h \sum_{i=0}^{k} \beta_i y'(t_{n+1-i}).$$

Записав разложение в ряд Тейлора в окрестности t_{n+1}, получаем

$$lte_n = \sum_{j=0}^{\infty} C_j h^j y^{(j)}(t_{n+1}), \tag{2.25}$$

где

$$C_0 = \sum_{i=0}^{k} \alpha_i, \qquad C_1 = -\sum_{i=0}^{k} [i\alpha_i + \beta_i],$$

$$C_j = (-1)^j \sum_{i=1}^{k} [\frac{i^j \alpha_i}{j!} + \frac{i^{j-1}\beta_i}{(j-1)!}], \; j = 2, 3, \ldots$$

В случае, когда рассматриваемый ЛММ сходится, его порядок равен p, если локальная погрешность формулы имеет порядок $O(h^{p+1})$. Из разложения (2.25) видно, что формула имеет порядок p, если $C_j = 0$ при $j = 0, 1, \ldots, p$. Кроме того,

$$lte_n = C_{p+1} h^{p+1(p+1)} y(t_{n+1}) + \ldots = C_{p+1} h^{p+1(p+1)} y(t_n) + \ldots \tag{2.26}$$

Для лучшего понимания обсуждаемых вопросов будет полезно рассмотреть два простых примера. Перепишем разложение в ряд Тейлора следующим образом

$$y(t_n) = y(t_{n+1}) - hy'(t_{n+1}) + \frac{h^2}{2} y''(t_{n+1}) + \ldots$$

С использованием этого равенства можно непосредственно доказать, что локальная погрешность формулы неявного метода Эйлера определяется равенством

$$lte_n = -\frac{h^2}{2} y''(t_n + h) + \ldots = -\frac{h^2}{2} y''(t_n) + \ldots \tag{2.27}$$

Читателю предлагается в качестве упражнения 2.6 показать, что локальная погрешность формулы для формулы трапеций АМ2 удовлетворяет равенству

$$lte_n = -\frac{h^3}{12} y'''(t_n) + \ldots \tag{2.28}$$

Рассмотрим некоторые важные детали доказательства сходимости. Предположим, что исследуемый явный ЛММ определяется формулой

$$y_{n+1} = y_n + h \sum_{i=1}^{k} \beta_i f(t_{n+1-i}, y_{n+1-i}).$$

Заметим, что этому классу численных методов принадлежат метод Адамса–Башфорта. Решение ОДУ удовлетворяет этому равенству с точностью до небольшого возмущения, равного локальной погрешности формулы

$$y(t_{n+1}) = y(t_n) + h \sum_{i=1}^{k} \beta_i f(t_{n+1-i}, y(t_{n+1-i})) + lte_n.$$

Вычитая первое равенство из второго, получаем

$$y(t_{n+1}) - y_{n+1} = y(t_n) - y_n + h\sum_{i=1}^{k}\beta_i[f(t_{n+1-i}, y(t_{n+1-i})) - f(t_{n+1-i}, y_{n+1-i}] + lte_n.$$

Если локальная погрешность формулы имеет порядок $O(h^{p+1})$, с использованием условия Липшица можно получить следующее неравенство

$$\|y(t_{n+1}) - y_{n+1}\| \leqslant \|y(t_n) - y_n\| + h\sum_{i=1}^{k}|\beta_i|L\|y(t_{n+1-i}) - y_{n+1-i}\| + Ch^{p+1},$$

которое включает информацию об ошибках, произошедших на шагах, предшествующих моменту времени t_n. Тогда с учетом равенства

$$E_m = \max_{j\leqslant m}\|y(t_j) - y_j\|,$$

можно получить следующую оценку

$$\|y(t_{n+1}) - y_{n+1}\| \leqslant (1 + h\mathcal{L})E_n + Ch^{p+1},$$

где

$$L\sum_{i=0}^{k}|\beta_i| = \mathcal{L}.$$

Тогда

$$E_{n+1} \leqslant (1 + h\mathcal{L})E_n + Ch^{p+1}.$$

Это выражение для оценки на рост ошибки при выполнении одного шага сходно с соответствующим выражением для одношаговых методов, полученным ранее. С использованием этого результата можно доказать, что ошибка на всем интервале $[a, b]$ имеет порядок $O(h^p)$. Аналогичные рассуждения позволяют доказать сходимость неявных методов, определяемых формулами Адамса–Моултона, а также сходимость методов, основанных на парах формул предиктор–корректор, например на паре AB-AM в форме PECE.

Ранее было показано, что сходимость метода непосредственно следует из доказательства ограниченности ошибки. Альтернативным способом доказательства сходимости является установление того факта, что малые возмущения численного решения не увеличиваются в процессе численного интегрирования до величины, пропорциональной $O(h^{-1})$. Поскольку для выполнения интегрирования на всем интервале от a до b необходимо выполнить $(b - a)h^{-1}$ шагов, сформулированное выше условие приблизительно соответствует требованию того, что на каждом отдельном шаге ошибка не более чем суммируется. Формула численного метода, отвечающая этому свойству называется *нуль-устойчивой* (zero-stable). В узлах сетки решение ОДУ удовлетворяет формуле с малым возмущением, равным локальной погрешности формулы, т. е. с точностью до величины порядка $O(h^{p+1})$. Тогда из нуль-устойчивости этой формулы следует, что в узлах сетки разность между решением ОДУ и соответствующим численным решением имеет порядок $O(h^p)$, т. е. рассматриваемый метод сходится и имеет порядок p.

Доказательство сходимости ЛММ общего вида начинается аналогично тому, как это делалось в рассмотренном выше случае, но с учетом нового обстоятельства, заключающегося в том, что накопленная ошибка не является простой суммой ошибок на предыдущих шагах, т. е. не равна величине кратной h. Это не только усложняет анализ, но и приводит к тому, что в случаях использования некоторых формул произошедшие на предыдущих шагах ошибки усиливаются, т. е. сходимость соответствующего метода нарушается. В классической теории ЛММ (см., например, [Henrici, 1962, 1977]) установлены необходимые и достаточные условия нуль-устойчивости формулы в терминах ее коэффициентов. Кроме того, показано, что точность нуль-устойчивых ЛММ не превышает половины порядка точности используемых в соответствующих формулах данных. Поэтому одной из причин для практического использования формул Адамса и формул, определяющих методы МДН, является то, что эти формулы являются нуль-устойчивыми и в вышеуказанном смысле обеспечивают максимально возможный порядок точности вычислений. Ограничение на порядок является необходимым условием устойчивости формулы, но не достаточным. В действительности, формулы, определяющие методы МДН порядка 7 и более, не являются нуль-устойчивыми.

В качестве примера линейного многошагового метода, не являющегося нуль-устойчивым, рассмотрим явную формулу третьего порядка [Isaacson & Keller, 1966, стр. 381]

$$y_{n+1} + \tfrac{3}{2}y_n - 3y_{n-1} + \tfrac{1}{2}y_{n-2} - 3hf(t_n, y_n) = 0. \tag{2.29}$$

Выполняя численные эксперименты, мы исследовали эту формулу в сравнении с AB3

$$y_{n+1} - y_n - h\left[\tfrac{23}{12}f(t_n, y_n) - \tfrac{16}{12}f(t_{n-1}, y_{n-1}) + \tfrac{5}{12}f(t_{n-2}, y_{n-2})\right] = 0. \tag{2.30}$$

Эти формулы имеют одинаковый порядок и в них используются одни и те же данные, поэтому сравнение их эффективности не является задачей с очевидным решением. Тем не менее из теории ЛММ известно, что при $h \to 0$ нуль-устойчивая формула AB3 является сходящейся, а формула (2.29) — нет. Цель проведенного численного эксперимента заключалась в том, чтобы исследовать зависимость максимальной величины ошибки от значения шага h при интегрировании ОДУ $y' = -y$, $y(0) = 1$ на интервале $[0, 1]$. Точные начальные значения были получены с использованием аналитического решения. В таблице 2.4 приведены максимальные значения ошибки при различной величине шага интегрирования $h = 2^{-i}, i = 2, 3, \ldots, 10$. Можно заметить, что AB3 сходится при $h \to 0$. Действительно, при уменьшении h вдвое максимальное значение ошибки уменьшается в 8 раз, что и следовало ожидать при использовании формулы третьего порядка. С другой стороны, численный эксперимент показывает, что формула (2.29) не является сходящейся. Соответствующая величина ошибки невелика при сравнительно больших значениях h, но это обусловлено тем, что процесс численного интегрирования, начинавшийся в изначально точном значении решения, прерывался после выполнения нескольких шагов вследствие того, что эффекты от накопления ошибки настолько усиливались, что точность вычисления численного решения становилась совершенно неприемлемой. В упражнении 2.8 читателю предлагается самостоятельно

Таблица 2.4: Максимальные ошибки при $h = 2^{-i}$

i	AB3	Формула (2.29)
2	1.34e−003	9.68e−004
3	2.31e−004	6.16e−003
4	3.15e−005	1.27e+000
5	4.08e−006	6.43e+005
6	5.18e−007	2.27e+018
7	6.53e−008	4.23e+044
8	8.19e−009	2.27e+098
9	1.03e−009	1.03e+207
10	1.28e−010	Бесконечность

выполнить для формулы (2.29) этот численный эксперимент, предварительно проверив, что она имеет порядок 3.

Если алгоритм численного интегрирования допускает переменный шаг, естественно предположить, что существуют некоторые ограничения на скорость изменения этой величины, при выполнении которых можно гарантировать устойчивость и сходимость метода при стремлении максимального значения шага интегрирования к нулю. Если относительное число шагов, выполненных с приемлемой точностью вычислений (успешных шагов), равномерно ограничено сверху, свойства сходимости и устойчивости для методов Адамса могут быть доказаны аналогично тому, как это было сделано в случае постоянного шага интегрирования [Shampine, 1994]. На практике это условие выполнено, поскольку численные процедуры лимитируют скорость увеличения шага интегрирования, что обусловлено причинами, объясненными в подпараграфе 2.2.1. В случае методов МДН ситуация несколько иная. Из классической теории ЛММ известно, что методы МДН порядка менее 7, применяемые при постоянных шагах интегрирования, являются устойчивыми и сходящимися при стремлении к нулю величины шага интегрирования. Программная реализация этих методов часто предполагает, что величина шага интегрирования остается постоянной на протяжении нескольких шагов и при необходимости изменяется на новое постоянное значение. Существуют теоретические результаты, из которых следует устойчивость и сходимость подобных реализаций при условии, что величина и частота указанных изменений шага интегрирования ограничены. К сожалению, эти теоретические результаты не позволяют в конкретной ситуации ответить на практические вопросы о допустимых значениях величины и частоты изменения шага.

Все численные методы, используемые на практике, являются устойчивыми при стремлении к нулю максимального значения шага интегрирования. Однако при численном исследовании специфических случаев может возникать вопрос: достаточно ли малым выбран шаг интегрирования, чтобы гарантировать устойчивость

численного интегрирования? Можно выписать выражения, характеризующие распространение в процессе интегрирования малых возмущений численного решения, но они имеют весьма сложный вид и не позволяют дать сколько-нибудь определенный ответ на сформулированный выше вопрос. Поэтому обратимся к стандартным методам анализа устойчивости рассматриваемого ОДУ. Основная идея заключается в том, чтобы аппроксимировать автономное уравнение $y' = f(y)$ в окрестности (t^*, y^*) линейным уравнением с постоянными коэффициентами

$$u' = f(y^*) + \frac{\partial f}{\partial y}(y^*)(u - y^*). \tag{2.31}$$

Это ОДУ является неустойчивым, если существуют решения, начинающиеся вблизи (t^*, y^*), которые быстро уходят на бесконечность. Разность двух решений этого линейного уравнения представляет собой решение однородного уравнения

$$v' = \frac{\partial f}{\partial y}(y^*)v.$$

Для простоты изложения предположим, что локальная матрица Якоби $\frac{\partial f}{\partial y}(y^*)$ диагонализируема, т. е. существует матрица T, составленная из собственных векторов этой матрицы Якоби, такая что $T^{-1}\frac{\partial f}{\partial y}(y^*)T = \operatorname{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_d\}$. При замене переменных $w = T^{-1}v$ уравнения разделяются: j-я компонента вектора w удовлетворяет $w_j' = \lambda_j w_j$. Таким образом, линейное однородное уравнение принимает следующий вид

$$w' = \lambda w, \tag{2.32}$$

где λ — диагональная матрица, на диагонали которой стоят собственные числа локальной матрицы Якоби. Далее это уравнение мы будем называть *тестовым*. Решения этого уравнения таковы, что если $\operatorname{Re}(\lambda_j) > 0$ для некоторого j, то $w_j(t)$ экспоненциально возрастает и, следовательно, аналогичным свойством обладает и $v(t)$. Поскольку разность между двумя решениями (2.31) экспоненциально возрастает, исходное ОДУ неустойчиво. С другой стороны, если $\operatorname{Re}(\lambda_j) \leqslant 0$ для всех j, то $w(t)$ и, следовательно, $v(t)$ ограничены. Таким образом, аппроксимирующее уравнение (2.31) устойчиво вблизи (t^*, y^*). Анализ стандартных численных методов может быть выполнен аналогичным образом и устойчивость исследуемого численного метода будет зависеть от устойчивости численного решения тестового уравнения. Как мы видели ранее, поведение численного решения подобного уравнения может быть легко исследовано при постоянном шаге интегрирования h и поэтому мы можем ответить на фундаментальный вопрос: если $\operatorname{Re}(\lambda_j) \leqslant 0$ для всех j, т. е. если аппроксимирующее уравнение устойчиво вблизи (t^*, y^*), насколько мал должен быть шаг h, чтобы численное решение было устойчивым? Теория *абсолютной устойчивости*, которую мы здесь подробно не рассматриваем, содержит множество практических результатов, полезных при понимании процесса численного интегрирования. Кроме того, в этой теории формулируются необходимые условия, которые могут быть использованы для тестирования численного метода: если они не выполнены для тестового уравнения, то рассматриваемый метод может найти лишь ограниченное применение при численном исследовании уравнений общего вида.

Использование теории абсолютной устойчивости мы продемонстрируем на примере метода Эйлера. Предположим, что $\mathrm{Re}(\lambda) \leqslant 0$, т. е. все решения тестового уравнения ограничены и, следовательно, уравнение устойчиво. Применяя явный метод Эйлера к этому уравнению, получаем

$$y_{n+1} = y_n + h\lambda y_n = (1 + h\lambda)y_n.$$

Легко видеть, что для ограниченности численного решения необходимо $|1 + h\lambda| \leqslant 1$. Множество

$$S = \{|1 + z| \leqslant 1, \mathrm{Re}(z) \leqslant 0\}$$

называется *областью (абсолютной) устойчивости* явного метода Эйлера. Если $h\lambda \in S$, численный метод называется устойчивым. Если $h\lambda$ не принадлежит S, численное решение уходит на бесконечность и процесс интегрирования ОДУ прекращается. Этот подход применим и к задачам более общего вида. В этом случае соответствующее условие заключается в том, что $h\lambda_j \in S$ для всех собственных значений λ_j локальной матрицы Якоби $\frac{\partial f}{\partial y}$. В теории абсолютной устойчивости установлены и другие необходимые условия устойчивости.

Как нам уже известно, явный метод Эйлера является устойчивым для любого значения λ, поскольку этот метод сходится. Заметим, что $L = |\lambda|$ — константа Липшица для тестового уравнения. В классической ситуации, когда величина $L(b - a)$ невелика, любое ограничение на размер шага интегрирования, вводимое для обеспечения устойчивости вычислительного процесса, не является ограничительным. Но что, если $\mathrm{Re}(\lambda) < 0$ и $|\lambda|(b-a) \gg 1$? В этом случае дифференциальное уравнение устойчиво (любые два решения экспоненциально сходятся друг к другу), но для устойчивости явного метода Эйлера необходимо, чтобы величина h не превышала $|\lambda|^{-1}$.

Выбор шага интегрирования определяется двумя основными требованиями: необходимостью обеспечения заданной точности вычислений и устойчивости вычислительного процесса. В общем случае размер шага интегрирования определяется точностью используемой формулы, но при исследовании жестких задач он может определяться также и требованиями устойчивости. Для лучшего понимания существа проблем, связанных с жесткими задачами, рассмотрим ЗНУ

$$y' = -100y + 10, \; y(0) = 1 \tag{2.33}$$

на интервале $0 \leqslant t \leqslant 10$. В упражнении 2.11 читателю предлагается определить наибольший шаг, при котором главный член в выражении для локальной погрешности формулы для явного метода Эйлера меньше по абсолютной величине заданного значения абсолютной ошибки. Качественные особенности вычислительного процесса при изменении величины шага интегрирования легко выявить с использованием аналитического решения рассматриваемого уравнения

$$y(t) = \frac{1}{10} + \frac{9}{10}e^{-100t},$$

из которого видно, что существует начальный период быстрого изменения решения, называемый *пограничным слоем* или *переходным процессом*. На этом отрезке

метод должен использовать малый шаг для обеспечения заданной точности численного решения. При столь малом шаге процесс интегрирования автоматически становится устойчивым и задача не является жесткой. По прошествии небольшого интервала времени решение $y(t)$ становится практически постоянным и поэтому на этом интервале заданную точность можно обеспечить при бóльших значениях шага интегрирования. Однако для обеспечения устойчивости вычислительного процесса эта величина должна удовлетворять неравенству $|1 + h(-100)| \leqslant 1$, что делает существенное увеличение шага невозможным. Действительно, если увеличить шаг более $h = 0.02$, численное решение уходит на бесконечность, несмотря на то, что при таком большом шаге это решение может быть легко аппроксимировано. К счастью современные численные процедуры, в которых предусмотрен автоматический выбор шага интегрирования, позволяют избежать этой ситуации: если шаг интегрирования достаточно мал для того, чтобы интегрирование было устойчивым, шаг увеличивается, но при условии, что обеспечивается заданная точность аппроксимации решения. Если при увеличении шага интегрирования превышен некоторый порог, за которым происходит нарушение устойчивости вычислительного процесса, ошибки начинают возрастать. Если величина ошибки превысила заданное допустимое значение, выполненный шаг рассматривается как неуспешный, величина шага интегрирования уменьшается и вычисления выполняются заново. Эта процедура повторяется до тех пор, пока шаг не становится достаточно малым для того, чтобы вычислительный процесс снова стал устойчивым. После этого весь логический цикл повторяется. Использование подобного подхода позволяет получить численное решение с заданной точностью, но вычислительные затраты очень высоки и это связано не только с использованием малого шага интегрирования, обеспечивающего устойчивость вычислительного процесса, но также с наличием большого количества неуспешных шагов. Пример 2.15 иллюстрирует вышеописанную процедуру.

Численный эксперимент с ЗНУ (2.33) весьма поучителен. Поскольку $0 < y(t) \leqslant \leqslant 1$, при допустимом значении относительной ошибки 10^{-12} и умеренном значении допустимого значения абсолютной ошибки в качестве критерия контроля ошибки вычислений фактически используется критерий контроля только абсолютной ошибки. В таблице 2.5 приведены результаты решения рассматриваемой ЗНУ с использованием явного метода Рунге–Кутта `ode45` при различных значениях допустимого значения абсолютной ошибки. Указанная численная процедура предназначена для решения нежестких задач, однако она может быть успешно использована и в этом случае, т. к. реализованный в ней контроль ошибки обеспечивает устойчивость вычислительного процесса. Кроме того, вычислительные затраты, необходимые для решения этой ЗНУ, оказались приемлемыми, т. к. рассматриваемая задача является лишь умеренно жесткой. Заметим однако, что имеется относительно большое число неуспешных шагов; число успешных шагов остается постоянным при уменьшении значения допустимой ошибки. Указанные особенности характерны для жестких задач, т. к. на протяжении большей части интервала интегрирования величина шага определяется, в первую очередь, не значением допустимой ошибки. Численная процедура `ode15s`, основанная на методе МДН и предназначенная для решения жестких задач, ведет себя совершенно по-другому. Число успешных шагов зависит от значения допустимой ошибки, т. к. размер шага

Таблица 2.5: Решение умеренно жесткой ЗНУ

Solver	AE	ME	SS	FS
`ode45`	1.0e−1	1.1e−1	303	26
	1.0e−2	1.1e−2	304	26
	1.0e−3	1.1e−3	307	19
	1.0e−4	1.1e−4	309	19
`ode15s`	1.0e−1	3.4e−2	23	0
	1.0e−2	8.2e−3	29	0
	1.0e−3	1.1e−3	39	0
	1.0e−4	1.6e−4	65	0

AE — допустимая абсолютная ошибка; ME — максимальная ошибка; SS — число успешных шагов; FS — число неуспешных шагов

определяется заданной точностью. При решении рассматриваемой задачи с использованием процедуры `ode15s` не произошло ни одного неудачного шага, в то время как при применении процедуры `ode45` количество неудачных шагов было немалым вследствие того, что эта процедура характеризуется конечной областью устойчивости.

Жесткие задачи представляют собой очень важный класс практических задач, и поэтому нам совершенно необходимо найти численные методы, которые были бы более устойчивыми по сравнению с явным методом Эйлера. К счастью, искомое решение этой проблемы не так глубоко запрятано, как могло бы показаться на первый взгляд. При решении тестового уравнения с использованием неявного метода Эйлера использовалась формула

$$y_{n+1} = y_n + h\lambda y_{n+1}.$$

Следовательно

$$y_{n+1} = \frac{1}{1 - h\lambda} y_n. \tag{2.34}$$

Тогда область устойчивости неявного метода Эйлера определяется равенством

$$S = \left\{ \left| \frac{1}{1 - z} \right| \leqslant 1, \mathrm{Re}(z) \leqslant 0 \right\},$$

т. е. представляет собой левую полуплоскость комплексной плоскости. Это свойство формулы называется *A-устойчивостью*. Таким образом, можно заметить, что в случае использования неявного метода Эйлера какие-либо ограничения на размер шага, призванные обеспечить устойчивость процесса интегрирования, отсутствуют. Однако следует отметить, что при выполнении анализа устойчивости

этого метода было выполнено множество аппроксимаций и поэтому не следует сильно доверять приведенному выше заключению. Тем не менее неявный метод Эйлера действительно демонстрирует великолепную устойчивость в большинстве ситуаций. Этот метод принадлежит к классу МДН1 и так же как и другие методы МДНs порядка от 2 до 6 включая, имеет область устойчивости, продолжимую до бесконечности. Только формулы порядка 1 и 2 являются A-устойчивыми. Остальные формулы устойчивы в секторе

$$\left\{z = re^{i\theta} | r > 0, \pi - \alpha < \theta < \pi + \alpha\right\},$$

содержащем всю отрицательную часть вещественной оси. Эта ограниченная версия свойства A-устойчивости называется *$A(\alpha)$-устойчивостью*. При увеличении порядка формулы угол α становится меньше и метод МДН7 становится неустойчивым при любых значениях угла α.

Подобно методу МДН1, формула трапеций является A-устойчивой. Пара формул предиктор–корректор, состоящая из предиктора в виде явной формулы Эйлера и формулы корректора, заданной формулой трапеций, является явной формулой Рунге–Кутта, построенной по методу Хойна. Нетрудно найти область устойчивости метода Хойна и показать, что она конечна. В действительности, все явные методы Рунге–Кутта имеют конечные области устойчивости. Следовательно, неявный метод и пара предиктор–корректор, в которой используется тот же неявный метод в качестве корректора, могут иметь важные качественные различия.

В упражнении 2.9 читателю предлагается определить область устойчивости для некоторых одношаговых методов и доказать утверждения, аналогичные сформулированным выше. В упражнении 2.16 предлагается определить области устойчивости для методов ЛММ. Напомним, что при доказательстве устойчивости обычно показывается, что возмущение численного решения не возрастает на последующих шагах. Однако если $\mathrm{Re}(\lambda) < 0$, возмущение в исследуемом ОДУ экспоненциально убывает. Было бы желательно, чтобы численный метод, предназначенный для численного решения подобного ОДУ, так же обладал бы соответствующим свойством. Из выражения (2.34) для МДН1 видно, что возмущения подавляются при $h\mathrm{Re}(\lambda) \ll -1$. $A(\alpha)$-устойчивая формула, характеризующаяся подобным свойством, называется *$L(\alpha)$-устойчивой*. Таким образом, сходящиеся МДН-методы являются $L(\alpha)$-устойчивыми. В случае формулы трапеций возмущения подавляются при $h\mathrm{Re}(\lambda) \ll -1$; эта формула A-устойчива, но не $L(\alpha)$-устойчива.

Интересно отметить, что методы МДН являются устойчивыми при *всех* достаточно больших $|h\lambda|$. Если $\mathrm{Re}(\lambda) > 0$ и МДН является устойчивым при $h\lambda$, то соответствующее решение тестового уравнения возрастает, а численное — убывает. Разумеется, численное решение обладает надлежащими качественными свойствами при всех достаточно малых $h\lambda$, т. к. указанные методы являются сходящимися, но если $\mathrm{Re}(\lambda) > 0$ и h слишком велико, эффект сильного подавления может оказаться нежелательным. Эти результаты иллюстрируются упражнениями 2.12 и 2.13. В упражнении 2.14 читателю предлагается исследовать вопрос о том, как некоторые особенности программной реализации численного метода могут быть связаны с его устойчивостью.

За описанными в предыдущем абзаце особенностями численных методов что-то скрывается. Но что? Зачем вообще вдаваться в тонкости численных методов интегрирования для простых нежестких задач? Все численные методы, характеризующиеся бесконечной областью устойчивости, являются неявными. Ранее, при исследовании неявного метода Эйлера AM1, мы обсуждали вопрос о вычислении соответствующей формулы методом простых итераций. К сожалению, ограничение на размер шага, обеспечивающее устойчивость метода простых итераций, не менее ограничительно, чем ограничения, обеспечивающие устойчивость явного метода. Для того, чтобы убедиться в этом, вычислим формулу для неявного метода Эйлера с использованием метода простых итераций в простейшей ситуации — при численном решении тестового уравнения. Итерация определяется равенствами

$$y_{n+1}^{[m+1]} = y_n + h\lambda y_{n+1}^{[m]}$$

и

$$|y_{n+1} - y_{n+1}^{[m+1]}| = |h\lambda|\,|y_{n+1} - y_{n+1}^{[m]}|.$$

Очевидно, для устойчивости этой процедуры необходимо $|h\lambda| < 1$. Это требование не столь ограничительно при решении нежестких задач, но при решении жестких задач оно становится неприемлемым. Например, в случае ЗНУ (2.33) для обеспечения устойчивости соответствующей численной процедуры необходимо потребовать, чтобы величина h была меньше 0.01. Таким образом, это ограничение на размер шага, обусловленное применением метода простых итераций при использовании неявного метода Эйлера, является более ограничительным, чем соответствующее ограничение для явного метода Эйлера, обусловленное необходимостью обеспечения устойчивости! При решении жестких задач мы должны использовать более мощные средства для решения алгебраических уравнений, связанных с применением неявного метода. В случае МДН эти уравнения имеют вид

$$y_{n+1} = h\gamma f(t_{n+1}, y_{n+1}) + \psi, \tag{2.35}$$

где γ — константа, характеризующая метод, и ψ содержит все члены, связанные с памятью алгоритма $y_n, y_{n-1}, \ldots$ Так же как и в случае использования метода простых итераций в нежестких задачах, в рассматриваемом случае важно иметь хорошую первоначальную оценку решения $y_{n+1}^{[0]}$. Она может быть получена с использованием интерполяционного полинома $Q(t)$, построенного по точкам $y_n, y_{n-1}, \ldots$ В этом случае искомая оценка определяется равенством $y_{n+1}^{[0]} = Q(t_{n+1})$. После этого для итеративного решения алгебраических уравнений может быть применен упрощенный метод итераций Ньютона (метод хорд). Линеаризованные уравнения имеют вид

$$y_{n+1}^{[m+1]} = \psi + h\gamma[f(t_{n+1}, y_{n+1}^{[m]}) + J(y_{n+1}^{[m+1]} - y_{n+1}^{[m]})],$$

где

$$J \approx \frac{\partial f}{\partial y}(t_{n+1}, y_{n+1}).$$

В этой форме записи хорошо видно использование линеаризованных уравнений, но очень важно на программном уровне эффективно реализовать соответствующие

вычисления. Последующая итерация должна вычисляться как коррекция Δ_m текущей итерации. Нетрудно получить следующие равенства

$$(I - h\gamma J)\Delta_m = \psi + h\gamma f(t_{n+1}, y_{n+1}^{[m]}) - y_{n+1}^{[m]}, \tag{2.36}$$

$$y_{n+1}^{[m+1]} = y_{n+1}^{[m]} + \Delta_m.$$

Итерационная матрица $I - h\gamma J$ является очень плохо обусловленной в случае, когда задача является жесткой. В упражнении 2.20 рассматривается подход, позволяющий модифицировать численную процедуру `ode15s` так, чтобы имелась возможность мониторинга состояния итерационной матрицы в процессе интегрирования. При решении очень плохо обусловленной линейной системы может оказаться, что верными являются лишь несколько первых цифр решения. Поэтому при непосредственном применении метода простых итераций очередное значение $y_{n+1}^{[m+1]}$ также будет содержать лишь несколько верных первых цифр. Однако, если воспользоваться методикой выполнения итераций посредством коррекций, то с каждой новой такой итерацией–коррекцией Δ_m искомое решение $y_{n+1}^{[m+1]}$ будет содержать все больше и больше верных первых цифр. Заметим, что на каждой итерации правая часть линейной системы (2.36) является мерой того, насколько хорошо полученное в результате очередной итерации решение удовлетворяет алгебраическим уравнениям (2.35). Зафиксировав значение аппроксимации J локальной матрицы Якоби, можно найти LU-разложение итерационной матрицы $I - h_n\gamma J$, с использованием которого можно эффективно найти решения для всех линейных систем, возникающих при применении метода простых итераций в момент t_{n+1}. Процедура выполнения LU-разложения матрицы $I - h_n\gamma J$ является вычислительно высокозатратной и поэтому в программных реализациях МДН шаг интегрирования обычно выбирается постоянным и на каждом из шагов используется одно и то же LU-разложение. В случаях, когда метод простых итераций сходится слишком медленно или если оказывается возможным использовать существенно больший шаг интегрирования (возможно, с заменой используемой формулы на формулу большего порядка), принимается решение о выполнении соответствующих модификаций численного алгоритма и вычислении новых значений итерационной матрицы и ее LU-разложения.

Вычисление неявной формулы путем решения уравнения упрощенным методом Ньютона может оказаться вычислительно ресурсоемкой задачей. При использовании этого подхода необходимо в первую очередь вычислить и сохранить в памяти компьютера аппроксимацию локальной матрицы Якоби J. После этого на каждой итерации необходимо решать линейную систему с матрицей $I - h\gamma J$. Получение аппроксимации матрицы Якоби является ресурсоемкой задачей и поэтому при программной реализации количество этих операций должно быть минимизировано. Новое значение итерационной матрицы вычисляется при существенном изменении шага интегрирования. В ранних версиях программных реализаций для жестких ЗНУ при изменении величины шага интегрирования вычислялось новое значение аппроксимации матрицы Якоби и оно сохранялось в памяти компьютера с удалением предыдущего значения матрицы Якоби, т. к. в то время объемы оперативной памяти компьютеров были невелики. Этот подход используется и в настоящее

время при решении очень больших систем ОДУ, но в некоторых современных программных реализациях этого метода, предназначенных для решения жестких ЗНУ, включая те, что используются в системе MATLAB, ранее вычисленные значения J сохраняются и заново используются при вычислении итерационной матрицы, когда процесс интегрирования протекает в установившемся режиме. Таким образом, при использовании такой численной процедуры для решения нежестких задач матрицы Якоби необходимо вычислить лишь несколько раз. Пакет линейной алгебры, входящий в состав среды программирования MATLAB, позволяет при решении нежестких задач умеренной размерности эффективно использовать численные процедуры, предназначенные для решения жестких задач.

Все численные процедуры для решения жестких ЗНУ имеют опции для вычисления аппроксимаций матриц Якоби конечными разностями, причем эта опция выбрана по умолчанию. Предположим, что нам необходимо вычислить аппроксимацию матрицы $\frac{\partial f}{\partial y}(t^*, y^*)$. Если через $e^{(j)}$ обозначить j-й столбец единичной матрицы, то вектор $y^* + \delta_j e^{(j)}$ представляет собой изменение j-й компоненты вектора y^* на величину δ_j. С использованием разложения в ряд Тейлора

$$f(t^*, y^* + \delta e^{(j)}) = f(t^*, y^*) + \frac{\partial f}{\partial y}(t^*, y^*)\delta_j e^{(j)} + O(\delta_j^2)$$

можно получить аппроксимацию j-го столбца матрицы Якоби:

$$\frac{\partial f}{\partial y}(t^*, y^*)e^{(j)} \approx \delta_j^{-1}[f(t^*, y^* + \delta e^{(j)}) - f(t^*, y^*)].$$

Если исследуемая система состоит из d уравнений, для вычисления аппроксимации матрицы Якоби с использованием этих формул нам потребуется d раз вычислить f. Представленный подход является стандартным способом вычисления матриц Якоби, но его программные реализации могут существенно различаться, т. к. выбор подходящего значения δ_j представляет собой сложную задачу. Эта величина должна быть достаточно малой для того, чтобы метод конечных разностей обеспечил достаточно точную аппроксимацию частных производных, и одновременно достаточно большой для того, чтобы эта аппроксимация не представляла собой лишь величину ошибки округления для данного компьютера. Если компоненты f сильно различаются по абсолютному значению, может даже оказаться, что невозможно найти такое значение δ_j, которое было бы подходящим в указанном смысле для всех компонент вектора частных производных. К счастью, нам требуется не абсолютно точное значение матрицы Якоби, а лишь такая ее аппроксимация, при которой обеспечивается сходимость упрощенного метода Ньютона. Некоторые численные алгоритмы, например процедура MATLAB `numjac`, учитывают при вычислении матриц Якоби существенную разницу абсолютных значений компонент функции f и в соответствии с этой информацией, а также на основе анализа ранее вычисленных значений аппроксимации матрицы Якоби осуществляют подстройку приращений δ_j. После этого вычисляется новая аппроксимация столбца матрицы Якоби при новом значении приращения. Численное определение значений матрицы Якоби является удобным методом, позволяющим получить

приемлемый результат, но численные методы интегрирования систем ОДУ становятся более робастными и быстрыми, если имеется возможность ввести в вычислительный процесс аналитическое представление матрицы Якоби.

Часто бывает, что большие системы ОДУ состоят из уравнений, каждое из которых зависит лишь от некоторых компонент вектора y. Если i-я компонента $f(t,y)$ не зависит от j-ой компоненты y, частная производная $\frac{\partial f_i}{\partial y_j}$ равна нулю. Если большинство элементов матрицы нулевые, то такая матрица называется *разреженной*. При использовании специальных методов представления разреженных матриц объем оперативной памяти компьютера, необходимой для хранения такой матрицы, можно уменьшить с d^2 до величины, кратной d. Если матрица Якоби является разреженной, то таковой является и соответствующая итерационная матрица. При этом вычислительная трудоемкость решения линейных систем с разреженной матрицей также может быть существенно снижена с учетом равенства нулю определенных элементов этой матрицы. Алгоритм, предложенный в работе [Curtis, Powell, & Reid, 1974], обеспечивает соответствующее снижение ресурсоемкости численной процедуры аппроксимации разреженной матрицы Якоби. Важным специальным случаем разреженных матриц Якоби является случай, когда ненулевые элементы расположены лишь на ее диагоналях. Например, если для всех i выполнено $J_{i,j} = 0$ для всех j, за исключением возможно $j = i-1, i$ и $i+1$, то соответствующая матрица J называется трехдиагональной или *ленточной* с шириной ленты 3. Если в матрице имеется m диагоналей, то вне зависимости от количества уравнений в рассматриваемой системе для вычисления аппроксимации столбца матрицы Якоби необходимо выполнить лишь m дополнительных вычислений функции f. Для того, чтобы воспользоваться всеми преимуществами работы с разреженными матрицами, требуются сравнительно небольшие затраты, и поэтому все современные численные методы интегрирования систем ОДУ приспособлены для использования ленточных матриц Якоби. Некоторые из них, включая те, что используются в MATLAB, могут работать с разреженными матрицами Якоби общего вида. Эти особенности алгоритмического характера имеют очень важное значение при численном исследовании больших систем. Более подробно эти вопросы рассмотрены в параграфе 2.3.3. Там же приведены несколько примеров.

С точки зрения затрат компьютерного времени и использования объемов оперативной памяти компьютера выполнение одного шага с использованием МДН, в котором для вычисления формул используется упрощенный итеративный метод Ньютона, является более затратной процедурой, чем выполнение одного шага с использованием любого другого метода, предназначенного для решения нежестких задач, и в котором вычисления формул осуществляются с использованием метода простых итераций или непосредственно. Таким образом, методы МДН, основанные на формулах, вычисляемых с использованием упрощенного итерационного метода Ньютона, имеют преимущество только в случаях, когда величина шага интегрирования в основном лимитируется требованием обеспечения устойчивости вычислительного процесса, т. е. когда исследуемая ЗНУ является жесткой. При решении нежестких задач формулы для методов МДН могут вычисляться с использованием метода простых итераций, но этот подход используется редко,

поскольку для подобных ситуаций можно использовать методы Адамса–Моултона, которые имеют аналогичную структуру алгоритма и обеспечивают существенно большую точность вычислений.

Следует отметить, что распознать жесткую задачу непросто. Мы уже убедились в том, что для решения жестких и нежестких задач применяются качественно различные численные методы. На практике задача определяется как жесткая, если наиболее эффективными методами ее численного решения оказываются численные методы, предназначенные для решения именно жестких задач. Для того, чтобы убедиться в этом, читателю предлагается выполнить упражнения 2.15, 2.18, 2.34 и 2.36. Эти упражнения позволяют также развить навык распознавания жестких задач. В жестких задачах существенное изменение значений решения ОДУ происходит на интервалах времени, длина которых очень мала по сравнению со всем интервалом времени, на котором выполняется интегрирование. Некоторые физические задачи могут быть сформулированы в терминах постоянных времени. При этом соответствующая жесткая задача должна иметь некоторую постоянную времени, которая мала по сравнению с интервалом времени, на котором проводится исследование, а само решение должно быть медленно меняющимся. Подобная ситуация рассматривается в упражнении 2.36. Большинство задач, которые описываются как жесткие, имеют промежутки времени, на протяжении которых решение быстро изменяется (например, пограничный слой, или переходный процесс). В действительности, на этих интервалах задача не является жесткой, т. к. численная процедура уменьшает шаг интегрирования просто для того, чтобы обеспечить заданную точность вычисления решения. Задача является жесткой, если вычисление решения с заданной точностью не представляет проблемы, но уменьшение шага обусловлено требованием обеспечения устойчивости вычислительного процесса. Выбор шага интегрирования таким, чтобы эта величина была подходящей и для начального переходного процесса и для гладкого установившегося режима, представляет собой фундаментальную проблему при решении жестких ЗНУ. Задача протонного переноса и пример Робертсона, рассмотренные в параграфе 1.4, могут служить иллюстрацией подобных задач.

Оценка ошибки и изменение порядка метода

Оценка величины локальной погрешности формулы при использовании линейных многошаговых методов в общем случае представляет собой более простую задачу по сравнению с оценкой величины локальной ошибки для методов Рунге–Кутта. Насколько легко получить оценку ошибки, настолько трудно доказать, что соответствующий критерий работоспособен на практике. Мы начнем наш анализ с получения оценки для локальной погрешности формулы. Один шаг по методу Адамса можно выполнить с использованием двух формул различного порядка. Действительно, ранее нами было установлено, что можно использовать предиктор AB($k-1$) с корректором AMk и получить пару формул порядка k. Так же как и в случае использования одношаговых методов, оценка ошибки вычисления по формуле AB($k-1$) порядка $k-1$ может быть получена путем сравнения результата, полученного с использованием этой формулы, с результатом, полученным с использованием упомянутой выше пары формул порядка k. Если управление численным алгоритмом осуществляется с учетом величины этой ошибки, если

интегрирование осуществляется с использованием пары предиктор–корректор, и если эта пара формул обеспечивает устойчивость вычислительного процесса и бóльшую точность решения, выполняется локальная экстраполяция. Описанный алгоритм используется в численной процедуре `ode113`. Другой подход основан на использовании разложения (2.26) для локальной погрешности формулы при постоянном шаге интегрирования. Можно заметить, что старшие члены в соответствующих выражениях для локальных погрешностей формул для методов Адамса–Башфорта и Адамса–Моултона одного порядка отличаются друг от друга постоянным множителем. Тогда для шага интегрирования, выполненного с использованием ABk и AMk, можно получить оценку локальной погрешности формулы, которая будет равна произведению указанного множителя на разность соответствующих двух результатов. Для продолжения процесса интегрирования используется более точный результат, полученный с использованием более устойчивого метода AMk. Ошибка вычислений по этой формуле входит в критерий контроля ошибки и поэтому локальная экстраполяция не производится. При таком подходе вычисление формул, соответствующих неявным методам Адамса–Моултона, осуществляется с заданной точностью с использованием метода простых итераций. Явная формула Адамса–Башфорта ABk используется для инициирования итеративного процесса, обеспечивающего вычисление AMk и оценки локальной погрешности формулы для AMk. На одном шаге итерации величины локальных погрешностей для формулы AMk и пары формул ABk – AMk равны с точностью до старших членов и поэтому аналогичный подход может быть использован при вычислении оценки локальной погрешности формулы для методов предиктор-корректор.

Для методов МДН естественным способом вычисления локальной погрешности формулы является непосредственная аппроксимация старшего члена в выражении для этой величины. Напомним, что для МДН1 это выражение имеет вид

$$lte_n = -\frac{h^2}{2}y''(t_n) + \ldots.$$

Поскольку в соответствующих формулах используются результаты численного дифференцирования, удобно выполнить интерполяцию по точкам y_{n+1}, y_n, и y_{n-1} и получить соответствующий квадратичный полином $Q(t)$. Тогда

$$est = -\frac{h^2}{2}Q''(t_n) \approx -\frac{h^2}{2}y''(t_n).$$

Именно с использованием этого равенства вычисляется оценка величины локальной погрешности формулы в численной процедуре `ode15s`. Аналогично, кубический интерполяционный полином используется в процедуре `ode23t` для аппроксимации значения производной в выражении для локальной погрешности формулы трапеций (2.28). Неявный метод Эйлера и формула трапеций являются одношаговыми методами, но решения, вычисленные на предыдущих шагах процесса численного интегрирования, используются при получении оценок локальной погрешности формулы. Также эти решения используются для предсказания решения алгебраических уравнений, которые должны быть решены для вычисления формул, соответствующих этим неявным методам. Таким образом, различия между неявными одношаговыми и многошаговыми методами на практике нивелируются.

В контексте вычисления оценок локальной погрешности формулы интересно отметить важный аспект, заключающийся в том, что оказывается возможным вычисление ошибки, возникающей при использовании формул различного порядка. Подобную ситуацию иллюстрируют примеры формул AM1 и AM2. Вспоминая процедуру вычисления оценки локальной погрешности формулы, можно понять, что при выполнении одного шага с использованием формулы второго порядка AM2 можно вычислить оценку локальной погрешности для формулы первого порядка AM1. Можно также предположить, что с использованием большего объема памяти алгоритма можно вычислить оценку этой ошибки при использовании формулы еще большего порядка AM3. С учетом этого результата открывается возможность адаптации порядка (используемой формулы) в процессе численного решения ЗНУ, позволяющей увеличить шаг интегрирования. Эта идея реализована в современных численных процедурах для методов Адамса и МДН, соответственно в `ode113` и `ode15s`. Имена этих функций содержат цифры, соответствующие порядку используемых формул. Например, порядок формул, которые используются в процедуре `ode113`, может варьировать от 1 до 13, а в процедуре `ode15s` — от 1 до 5. (Буква «s» в имени `ode15s` указывает на то, что эта процедура предназначена для решения жестких (stiff) задач.) Среди программных реализаций метода Адамса–Моултона и методов МДН следует отметить процедуру переменного порядка DIFSUB [Gear, 1971] и процедуру VODE [Brown et al., 1989]. Изменение порядка используемой формулы играет в программных реализациях методов Адамса и МДН и другую роль. Формулы наименьшего порядка являются одношаговыми и поэтому они используются при выполнении первого шага процесса интегрирования. На каждом очередном шаге вычисляется очередное значение решения, которое записывается в память алгоритма. После этого алгоритм выявляет возможность увеличения порядка используемой формулы с целью последующего увеличения величины возможного шага интегрирования. На практике порядок используемой формулы быстро возрастает до величины, наиболее адекватной получаемому решению. Выполнение начальных этапов численного интегрирования в соответствии с описанным алгоритмом является наиболее удобным и эффективным для численных процедур, в которых реализована возможность изменения шага и порядка (variable-step size, variable-order (VSVO)), и поэтому все основанные на методах Адамса и МДН численные процедуры реализованы так, как описано выше.

Получение выражений для оценок величины локальной погрешности формул для многошаговых методов является сравнительно простой задачей, но теоретическое обоснование полученных результатов сопряжено с определенными трудностями. Проблема заключается в том, что при получении формулы оценки мы неявно предполагали, что найденные на предыдущих шагах численные значения решения и его производных равны соответственно значениям точного решения $y(t_n), y(t_{n-1}), \ldots$ и его производных $y'(t_n), y'(t_{n-1}), \ldots$ Однако на практике эти ранее вычисленные значения содержат ошибки, величина которых может быть сравнима или даже больше величины ошибки, которую мы хотим оценить. Эта проблема особенно отчетливо проявляется, если внимательно проанализировать процедуру получения оценки локальной погрешности для формулы, порядок которой превосходит порядок формул(ы), используемых(ой) при вычислении значений, записываемых в память алгоритма. Обоснование формулы оценки в реальных

ситуациях является сложной задачей в частности потому, что сделать это бывает невозможно, если поведение ошибки не обладает некоторой известной регулярностью. Поэтому до сих пор мы обсуждали лишь порядок малости ошибки. Из классической теории линейных многошаговых методов известно, что при выполнении некоторых предположений ошибка ведет себя регулярно и тогда эта особенность может быть использована для обоснования формулы ошибки (см. например работу [Henrici, 1962] и более общий результат [Stetter, 1973]). Эти результаты не могут быть непосредственно перенесены на современные методы Адамса и МДН переменного порядка, поскольку они получены в предположении, что в процессе численного интегрирования всегда используется одна и та же формула и что изменение шага интегрирования (если таковое имеет место) обладает некоторой заранее известной регулярностью. Известны некоторые результаты о поведении ошибки в ситуациях, когда порядок (используемая формула) изменяется в процессе выполнения процедуры интегрирования. Теоретические результаты для случая постоянного порядка проливают некоторый свет на случай переменного порядка и существуют некоторые результаты (см. например [Shampine, 2002]), которые могут быть непосредственно применены в подобных ситуациях, но, к сожалению, следует отметить, что уровень наших знаний о поведении ошибки при использовании современных реализаций методов Адамса и МДН переменного порядка, далек от желаемого.

Непрерывное обобщение численных методов

Методы Адамса и МДН основаны на использовании интерполяционных полиномов, которые можно использовать при построении непрерывных обобщений этих методов. Если непрерывное обобщение выполняется в отношении многошаговых методов, то появляется возможность его использования при построении алгоритмов изменения величины шага интегрирования. Выше было показано, что при решении жестких ЗНУ имеются практические причины использовать постоянное значение шага интегрирования на протяжении нескольких шагов. В подобных ситуациях одним из самых простых способов изменения в момент t_n шага интегрирования на другое значение H является аппроксимация значения решения в соответствующий момент времени с использованием интерполяционного полинома, построенного по значениям решения в моменты $t_n, t_n - H, t_n - 2H, \ldots$ При наличии этих значений решения в узлах сетки с постоянным шагом H, начиная с момента t_n, можно использовать формулу с новым постоянным шагом H. Этот метод является стандартным при использовании методов МДН и иногда применяется в некоторых программных реализациях метода Адамса. Примером подобных реализаций является процедура DIFSUB.

■ УПРАЖНЕНИЕ 2.5

Для лучшего понимания существа методов Адамса и МДН:

- Выведите формулы AB2.
- Выведите формулы МДН2.

■ УПРАЖНЕНИЕ 2.6

Покажите, что локальная погрешность формулы AM2 (формула трапеций) определяется равенством

$$lte_n = -\frac{h^3}{12}y^{(3)}(t_n) + \ldots$$

■ УПРАЖНЕНИЕ 2.7

Покажите, что при численном решении уравнения $y' = -y$ с использованием AM2 (формула трапеций) ошибка при выполнении одного шага h из точки $(t_n, y(t_n))$ имеет следующий вид

$$y(t_n + h) - y_{n+1} = \frac{1}{12}h^3 y(t_n) + \ldots$$

(Это выражение определяет локальную погрешность формулы при решении указанного ОДУ с использованием AM2.) Покажите, что выражение для соответствующей ошибки при использовании AB1-AM2 в форме PECE (метод Хойна) имеет следующий вид

$$y(t_n + h) - y_{n+1} = -\frac{1}{6}h^3 y(t_n) + \ldots$$

Очевидно, что точность метода предиктор–корректор может отличаться от точности вычисления формулы корректора, выполняемого в соответствии со стандартной процедурой вычисления неявной формулы.

■ УПРАЖНЕНИЕ 2.8

Покажите, что формула (2.29) имеет порядок 3. Выполните численный эксперимент, описанный в тексте, и результаты которого приведены в таблице 2.4.

■ УПРАЖНЕНИЕ 2.9

Для лучшего понимания термина «область устойчивости»:

- Покажите, что область устойчивости неявного метода Эйлера содержит левую половину комплексной плоскости;
- Покажите, что областью устойчивости формулы трапеций является левая половина комплексной плоскости.
- Метод Хойна может рассматриваться как метод предиктор–корректор, построенный с использованием формулы предиктора, основанной на методе Эйлера, и корректора, заданного формулой трапеций. Покажите, что область устойчивости метода Хойна является конечным множеством.

■ УПРАЖНЕНИЕ 2.10

Для лучшего понимания программной реализации различных методов численного интегрирования ОДУ в среде MATLAB напишите программу для решения системы

$$y' = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} y, \; y(0) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

на интервале $[0, 1]$ при постоянном шаге интегрирования $h = 0.1$. Выведите график численного и аналитического решений.

- Решите эту ЗНУ с использованием метода Хойна. Для этого напишите функцию вида `[tnp1,ynp1] = Heun(tn,yn,h,f)`. Входными параметрами этой функции являются текущее значение решения (t_n, y_n), шаг интегрирования h и идентификатор функции f, вычисляющей правую часть исследуемого ОДУ. В результате на каждом шаге вы будете получать значение решения в очередной точке (t_{n+1}, y_{n+1}). В MATLAB реализована возможность вычисления функций, которые передаются в другую функцию в виде входного аргумента. Это осуществляется с использованием встроенной функции `feval`, например, для выполнения одного шага с использованием метода Эйлера следует написать следующий программный код

```
function [tnp1,ynp1] = Euler(tn,yn,h,f)
yp = feval(f,tn,yn);
ynp1 = yn + h*yp;
tnp1 = tn + h;
```

  Более детальная информация о функции `feval` приведена в документации MATLAB.
- Решите ЗНУ с использованием формулы трапеций, вычисляемой по методу простых итераций. Метод Хойна может рассматриваться как пара формул предиктор–корректор, в которой в качестве предиктора используется метод Эйлера, а в качестве корректора — коррекция по формуле трапеций. Модифицируйте функцию `Heun` и получите в результате функцию `AM2si`, с использованием которой итеративно вычисляется формула трапеций. Вообще говоря, выработать критерий остановки соответствующего итеративного процесса непросто, но в этом примере предлагается сделать следующее: если `p` и `c` — это текущее и новое значения искомого решения, то итеративный процесс должен быть остановлен, если выполнено условие `norm(c - p) < 1e-3*norm(c)`. Если этот критерий оказался невыполненным после десяти итераций, следует прекратить выполнение программы и вывести сообщение об ошибке.
- Эта ЗНУ не является жесткой, но мы рекомендуем решить ее с использованием формулы трапеций, чтобы получить некоторый опыт интегрирования жестких ЗНУ. Модифицируйте функцию `AM2si` следующим образом: `[tnp1,ynp1] = AM2ni(tn,yn,h,f,dfdy)`. Здесь `dfdy` — идентификатор функции $dfdy(t,y)$, вычисляющей матрицу Якоби $\frac{\partial f}{\partial y}(t,y)$. Перед вызовом функции `AM2si` используйте `feval` для вычисления (постоянной) матрицы Якоби J, вычислите итерационную матрицу $M = I - 0.5hJ$ и соответствующее LU-разложение, с использованием которого выполняются итерации. Как было объяснено в тексте, программный код итеративного процесса должен быть написан так, чтобы очередное итеративное значение `c` эффективно вычислялось в виде суммы текущего итеративного значения `p` и коррекции Δ_m. Критерием останова итеративного процесса является следующее условие: величина $\|\Delta_m\|$ должна быть меньше `1e-3*norm (c)`. Если этот критерий оказался невыполненным после десяти итераций, следует прекратить выполнение программы и вывести сообщение об ошибке. (Метод Ньютона очень быстро сходится для рассматриваемого ОДУ, но поскольку программа должна

сохранять свою работоспособность в общем случае, следует предусмотреть возможность автоматического прекращения работы программы.)

■ УПРАЖНЕНИЕ 2.11

В тексте этого раздела обсуждалось решение жесткой ЗНУ

$$y' = -100y + 10,\ y(0) = 1$$

на интервале $0 \leqslant t \leqslant 10$ с использованием AB1. Было установлено, что на начальном отрезке этого интервала необходимо использовать малый шаг интегрирования для того, чтобы вычислить решение на пограничном слое с заданной точностью, а на оставшейся части интервала, где решение остается почти постоянным, следует использовать сравнительно большой шаг интегрирования. Обоснуйте это утверждение, найдя максимальное значение шага интегрирования h, при котором абсолютная величина старшего члена в выражении для локальной погрешности формулы в точке t_n не превосходит допустимое значение абсолютной ошибки τ.

■ УПРАЖНЕНИЕ 2.12

Для удобства и простоты написания программного кода при применении неявного метода Эйлера пользователи часто используют постоянное значение шага интегрирования. Эта формула обладает очень хорошими свойствами устойчивости, но злоупотребление этим преимуществом может привести к тому, что станут неверными даже качественные характеристики полученного численного решения. В качестве иллюстрации этого утверждения рассмотрите ЗНУ

$$y' = 10y,\ y(0) = 1$$

и найдите ее численное решение с использованием неявного метода Эйлера при постоянном шаге интегрирования $h = 1$. Прокомментируйте результаты сравнения полученного численного решения с аналитическим решением $y(t) = e^{10t}$.

■ УПРАЖНЕНИЕ 2.13

Рассмотренные в упражнении 2.12 проблемы могут быть также проиллюстрированы на примере следующей ЗНУ

$$y' = -\frac{1}{t^2} + 10\left(y - \frac{1}{t}\right),\ y(1) = 1.$$

Если из физических соображений известно, что решение должно стремиться к нулю, можно ожидать, что оно может быть легко получено с использованием неявного метода Эйлера при постоянном шаге интегрирования. Заметим, что аналитическое решение имеет вид $y(t) = t^{-1}$ и оно действительно стремится к нулю. Результаты численного интегрирования при $h = 1$ приведены в таблице 2.6 и, как легко видеть, вполне приемлемы.

Найдите решение этой же ЗНУ с использованием численных процедур `ode45` и `ode15s` с переменным шагом интегрирования. Полученные численные решения должны быть абсолютно неправдоподобны. Почему указанные численные процедуры оказались непригодными в этом случае? Что можно сказать о рассматриваемой ЗНУ после анализа полученных результатов?

Таблица 2.6: Решение, полученное с использованием неявного метода Эйлера

t	y
2.0	0.472
3.0	0.330
4.0	0.248
5.0	0.199
6.0	0.166
7.0	0.142
8.0	0.124
9.0	0.110
10.0	0.099

Трудности, с которыми мы столкнулись при рассмотрении этого примера, не всегда могут быть легко преодолены в практических ситуациях. В подпараграфе 2.3.3 исследуется вопрос о том, как решение дифференциального уравнения в частных производных (ДУЧП) может быть аппроксимировано решением некоторой системы обыкновенных дифференциальных уравнений. Нахождение решения подобных задач всегда содержит элемент искусства. При использовании метода конечных разностей для аппроксимации некоторых ДУЧП может оказаться, что «очевидная» система ОДУ является неустойчивой. Иногда возможно демпфировать эту неустойчивость путем использования при нахождении решения неявного метода Эйлера с «большим» шагом интегрирования, однако при этом остается открытым вопрос о достоверности полученного численного решения, т. е. о том, насколько точно полученное численное решение системы ОДУ моделирует искомое решение рассматриваемого ДУЧП.

■ УПРАЖНЕНИЕ 2.14

Неявный метод Эйлера может оказаться неприменимым и в других прикладных задачах. Хорошо известно, что следует внимательно следить за тем, чтобы используемый неявный метод сохранял свою устойчивость, но в целях снижения вычислительных затрат при реализации алгоритма предиктор–корректор применяется только одна коррекция. Невозможно одновременно удовлетворить этим двум целям, т. к. увеличение устойчивости метода может быть выполнено за счет увеличения точности вычислений, т. е. вычислительных затрат. В качестве иллюстрации этого утверждения докажите конечность области устойчивости для пары предиктор–корректор, в состав которой входит явный метод Эйлера и одна коррекция, выполненная с использованием формулы неявного метода Эйлера. Эффект от использования всего лишь одной коррекции становится более явным, если показать, что пара предиктор–корректор не является устойчивой, например, при

$z = h\lambda = -2$, в то время как неявный метод Эйлера остается устойчивым при всех z, принадлежащих левой полуплоскости комплексной плоскости.

◘ УПРАЖНЕНИЕ 2.15

С теоретической и практической точек зрения это упражнение позволяет выявить существо свойства жесткости ЗНУ. Рассмотрим модель концентрации реагента $y(t)$ в процессе горения [O'Malley, 1991]

$$y' = f(y) = y^2(1 - y),$$

интегрирование которой следует выполнить на интервале $[0, 2\varepsilon^{-1}]$ и при начальных данных $y(0) = \varepsilon$. При анализе решений этой системы применялся метод возмущений. Для этого состояние системы до момента зажигания рассматривалось с малыми возмущениями $\varepsilon > 0$. Мы не будем приводить здесь полученные автором теоретические результаты и выполним численное интегрирование указанной ЗНУ при $\varepsilon = 10^{-4}$ с использованием основанной на методе МДН процедуры `ode15s`. Текст соответствующей программы приведен ниже

```
function ignition
epsilon = 1e-4;
options = odeset('Stats','on');
ode15s(@ode,[0, 2/epsilon],epsilon,options);
%================================
function dydt = ode(t,y)
dydt = y^2*(1 - y);
```

Эта программа позволяет графически вывести решение на экран и отобразить некоторую статистическую информацию в конце своей работы. Запустите эту программу на выполнение и убедитесь, что решение с начальным значением ε медленно увеличивается. В момент времени порядка $O(\varepsilon^{-1})$ реагент зажигается, и решение быстро достигает значения 1. Это увеличение происходит на интервале времени длиной $O(1)$. На остальной части интервала интегрирования решение остается практически постоянным и очень близким к предельному значению 1. Модифицируйте приведенную выше программу так, чтобы в ней использовалась численная процедура `ode45`, основанная на явном методе Рунге–Кутта. Для этого просто измените в тексте программы имя вызываемой процедуры. Запустите модифицированную программу на выполнение и убедитесь, что процесс численного интегрирования существенно замедляется после зажигания реагента несмотря на то, что к этому моменту решение остается практически постоянным. Для того, чтобы оценить временные характеристики выполненных вычислений, используйте функции `tic` и `toc`, позволяющие определить время, затраченное указанными процедурами для интегрирования системы на всем интервале и на первой его половине. При этом не следует выводить на экран получаемые в процессе интегрирования значения решения. Для этого замените строку с вызовом процедуры `ode15s` на следующую

```
[t,y] = ode15s(@ode,[0, 2/epsilon],epsilon,options);
```

Сравнив результаты, вы убедитесь в том, что на первой половине интервала интегрирования численная процедура для нежестких задач `ode45` работает быстрее вследствие того, что формулы, на которых она основана, более точны. Интересно отметить, что этот результат получен для случая, когда вычислительные затраты, связанные с вызовом в `ode15s` процедур линейной алгебры, минимальны (т. к. рассматриваемое ОДУ является скалярным). С другой стороны, на всем интервале интегрирования быстрее завершилась численная процедура для жестких задач `ode15s`, т. к. при использовании процедуры `ode45`, основанной на явных формулах Рунге–Кутта, на первой половине интервала интегрирования применялся достаточно малый шаг интегрирования для того, чтобы обеспечить устойчивость вычислительного процесса, в то время как при использовании основанной на методах МДН процедуры `ode15s` необходимость в этом не возникает. Статистика вычислительного процесса показала, что при использовании метода Рунге–Кутта на второй половине интервала интегрирования имело место множество неуспешных шагов, что типично для случаев применения численного метода с конечной областью устойчивости для решения жестких задач.

Для лучшего понимания полученных численных результатов выпишите матрицу Якоби $\frac{\partial f}{\partial y}$. Поскольку рассматриваемый случай является скалярным, единственным собственным значением этой матрицы является значение единственного элемента этой матрицы. Определите константу Липшица для $0 \leqslant y \leqslant 1$. Убедитесь в том, что ее значение невелико и поэтому рассматриваемая ЗНУ может быть жесткой лишь на больших интервалах. На начальной части интервала интегрирования решение является положительным, медленно меняется и имеет порядок $O(\varepsilon)$. Покажите с использованием линейной теории устойчивости и с учетом знака собственного значения, что ЗНУ является неустойчивой и, следовательно, не является жесткой на этой части интервала интегрирования. Покажите, что эта задача является лишь умеренно неустойчивой на этом интервале длиной $O(\varepsilon^{-1})$, вследствие чего ее решение может быть вычислено достаточно точно. В момент зажигания изменение значений решения происходит очень быстро и поэтому график на интервале $[0, 2\varepsilon^{-1}]$ может быть недостаточно детальным, однако с использованием `zoom` можно убедиться, что бóльшая часть изменений происходит на промежутке $[9650, 9680]$. Покажите, что на этом интервале и на всех других интервалах длиной $O(1)$ рассматриваемая ЗНУ не является жесткой. На второй половине интервала интегрирования решение близко к 1 и медленно меняется. Покажите с учетом знака собственного значения, что эта ЗНУ является устойчивой на этом интервале, а также то, что она является жесткой на интервале длиной $O(\varepsilon^{-1})$.

■ УПРАЖНЕНИЕ 2.16

Граница области устойчивости для линейного многошагового метода может быть определена с использованием *метода корневого годографа* (root locus method). Применение ЛММ с постоянным шагом h к тестовому уравнению $y' = \lambda y$ приводит к получению уравнения

$$\sum_{i=0}^{k} \alpha_i y_{n+1-i} - h \sum_{i=0}^{k} \beta_i (\lambda y_{n+1-i}) = \sum_{i=0}^{k} (\alpha_i - h\lambda\beta_i) y_{n+1-i} = 0,$$

являющегося линейным разностным уравнением порядка k с постоянными коэффициентами. Это уравнение может быть исследовано с использованием методов, аналогичных тем, что применимы к линейным ОДУ с постоянными коэффициентами порядка k. Каждый корень r характеристического полинома

$$\sum_{i=0}^{k}(\alpha_i - h\lambda\beta_i)r^{k-i}$$

представляет собой решение разностного уравнения вида $y_m = r^m$, $m = 0, 1, 2, \ldots$ Случай, когда r является кратным корнем, рассматривается аналогично тому, как это делается в случае исследования ОДУ. Кроме того, можно показать, что для заданного $z = h\lambda$ все решения разностного уравнения ограничены и, следовательно, линейный многошаговый метод устойчив, если все корни характеристического полинома по абсолютному значению меньше 1. Если точка z принадлежит границе области устойчивости, абсолютное значение корня r равно 1. Это утверждение можно использовать в качестве критерия для определения границы области устойчивости. Эта граница может быть изображена графически, если вывести на экран все точки z, для которых r является корнем с абсолютным значением 1. Для этого удобно переписать характеристический полином в виде $\rho(r) - z\sigma(r)$, где

$$\rho(r) = \sum_{i=0}^{k}\alpha_i r^{k-i}, \qquad \sigma(r) = \sum_{i=0}^{k}\beta_i r^{k-i}.$$

Тогда границей области устойчивости является кривая

$$z = \frac{\rho(r)}{\sigma(r)},$$

где $r = e^{i\theta}$ при θ, изменяющемся от 0 до 2π. Построение графиков границ области устойчивости для различных линейных многошаговых методов, например для явного метода Эйлера (АВ1), неявного метода Эйлера (АМ1, МДН1), формулы трапеций (АМ2) и МДН2, обсуждалось и будет обсуждаться в соответствующих местах текста этой книги. Кроме того, мы рекомендуем читателю рассмотреть другие формулы для линейных многошаговых методов, например, АМ3

$$y_{n+1} = y_n + h\left[\frac{5}{12}y'_{n+1} + \frac{8}{12}y'_n - \frac{1}{12}y'_{n-1}\right]$$

и МДН3

$$y_{n+1} = \frac{18}{11}y_n - \frac{9}{11}y_{n-1} + \frac{2}{11}y_{n-2} + \frac{6}{11}h'y_{n+1}.$$

Метод корневого годографа позволяет определить лишь границу области устойчивости. Однако методы МДН имеют неограниченные области устойчивости и поэтому они устойчивы *вне кривой*, которую можно изобразить графически. В этом смысле формула трапеций несколько отличается, т. к. она устойчива в левой половине комплексной плоскости (в упражнении 2.9 читателю предлагается доказать это утверждение). Другие упомянутые выше методы имеют конечные области устойчивости, и поэтому они устойчивы внутри указанной кривой.

§ 2.3. РЕШЕНИЕ ЗНУ С ИСПОЛЬЗОВАНИЕМ MATLAB

В простейшем случае при использования численных процедур MATLAB необходимо определить рассматриваемую ЗНУ, задав аргументы соответствующей процедуры. Это означает, что вы должны написать функцию, которая вычисляет правую часть исследуемого ОДУ $f(t,y)$, указать интервал интегрирования и вектор начальных условий. В MATLAB реализованы множество различных численных методов, но их интерфейс с пользователем унифицирован. В документации рекомендуется использовать процедуру `ode45`, если вы полагаете, что исследуемая задача не является жесткой, в противном случае используйте `ode15s`. Понятие жесткости обсуждалось в предыдущих параграфах, но с практической точки важно понимать, что если `ode15s` решает ЗНУ существенно быстрее, чем `ode45`, то эта задача является жесткой. Многие полагают, что если с использованием процедур типа `ode45` ЗНУ решается со значительными вычислительными затратами, то эта задача является жесткой. Это суждение ошибочно, т. к. значительные вычислительные затраты могут быть обусловлены другими причинами. Более того, эти же причины могут привести к аналогичным проблемам при использовании процедуры `ode15s`, предназначенной для решения жестких задач. С другой стороны, если в указанном смысле рассматриваемая задача решается процедурой `ode45` с трудом, а с использованием `ode15s` легко, то можно с большой уверенностью утверждать, что она является жесткой.

Существуют также чисто программистские аспекты, которые также важно обсудить. Программы MATLAB могут быть написаны в виде отдельных файлов (файлов сценария — script files) или в виде функций. В этой книге мы предпочитаем приводить примеры в виде функций, т. к. в этом случае можно использовать дополнительные вспомогательные функции (подфункции). Кроме того, многим пользователям представляется удобным иметь в файле, содержащим текст основной программы функцию, определяющую исследуемое ОДУ. Этот подход становится еще более актуальным в ситуациях, когда могут использоваться несколько функций, например, когда в решении ЗНУ осуществляется локализация наступления определенных событий (event location). Более того, применение соответствующих численных процедур для решения ЗГУ и ДУЗА возможно лишь при использовании в качестве входных аргументов функций, определяющих рассматриваемое дифференциальное уравнение.

ПРИМЕР 2.3.1

В главе 1 исследовались аналитические решения простых дифференциальных уравнений. (В упражнении 2.17 предлагалось решить эти уравнения численно.) В частности, рассматривалось ОДУ

$$y' = y^2 + t^2.$$

Выражение для общего решения этого уравнения оказалось достаточно сложным и включало функции Бесселя. Тем не менее, это уравнение достаточно просто и его решение может быть найдено в MATLAB аналитически, а также вычислено при начальном значении $y(0) = 0$ и графически изображено на экране компьютера на промежутке $0 \leqslant t \leqslant 1$ с использованием следующей программы

```
>> y = dsolve('Dy = y^2 + t^2','y(0) = 0')
>> ezplot(y,[0,1])
```

Подобный способ задания ОДУ приемлем лишь в простейших случаях и неудобен, когда исследуемая система является большой и сложной. Численные процедуры MATLAB позволяют использовать в качестве своих аргументов функции, в которых вычисляется правая часть рассматриваемого дифференциального уравнения. Например, для рассматриваемого случая эта функция имеет следующий вид

```
function dydt = f(t,y)
dydt = y^2 + t^2;
```

Этот текст должен быть записан в файл `f.m`. Тогда для численного решения этого уравнения на интервале $[0,1]$ и графического отображения решения может быть использована команда

```
[t,y] = ode45(@f,[0,1],0);
plot(t,y)
```

которая может быть записана либо в отдельный файл, либо реализована в виде функции. Разумеется, столь короткий текст можно просто набрать в командной строке. Первым аргументом функции `ode45` является указатель на функцию `@f`, в которой вычисляется правая часть ОДУ. Второй аргумент — интервал, на котором выполняется интегрирование $[a,b]$ и третий аргумент — начальное значение. Численная процедура вычисляет аппроксимации решения $y_n \approx y(t_n)$ в точках сетки

$$a = t_0 < t_1 < \ldots < t_N = b,$$

выбранных в соответствии с алгоритмом, заложенным в численной процедуре. Выходными значениями процедуры являются точки сетки, записанные в массив `t`, и соответствующие значения решения, записанные в массив `y`. Указанные выходные значения формируются в виде единого двумерного массива. Эта форма вывода удобна для последующего вызова процедуры построения графика решения на экране компьютера. На рисунке 2.1 представлены результаты вычислений. В результате работы этой программы (записанной в отдельном файле или набранной в командной строке) мы получаем массивы точек сетки `t` и соответствующих значений решения `y`, которые могут быть использованы, например, для отображения графиков отдельных компонент решения или для вывода решения в логарифмической шкале.

Все указанные входные аргументы являются необходимыми для определения рассматриваемой ЗНУ. Единственным аргументом, для которого предусмотрена возможность задания по умолчанию, является интервал интегрирования. Однако, как мы убедились ранее, устойчивость ЗНУ имеет фундаментальное значение для получения численного решения и зависит от длины этого интервала, а также от направления интегрирования. В системе MATLAB численные процедуры реализованы таким образом, чтобы избежать использования длинных списков входных аргументов, что характерно для численных процедур, предназначенных для использования в предварительно скомпилированных модулях. Например, в MATLAB

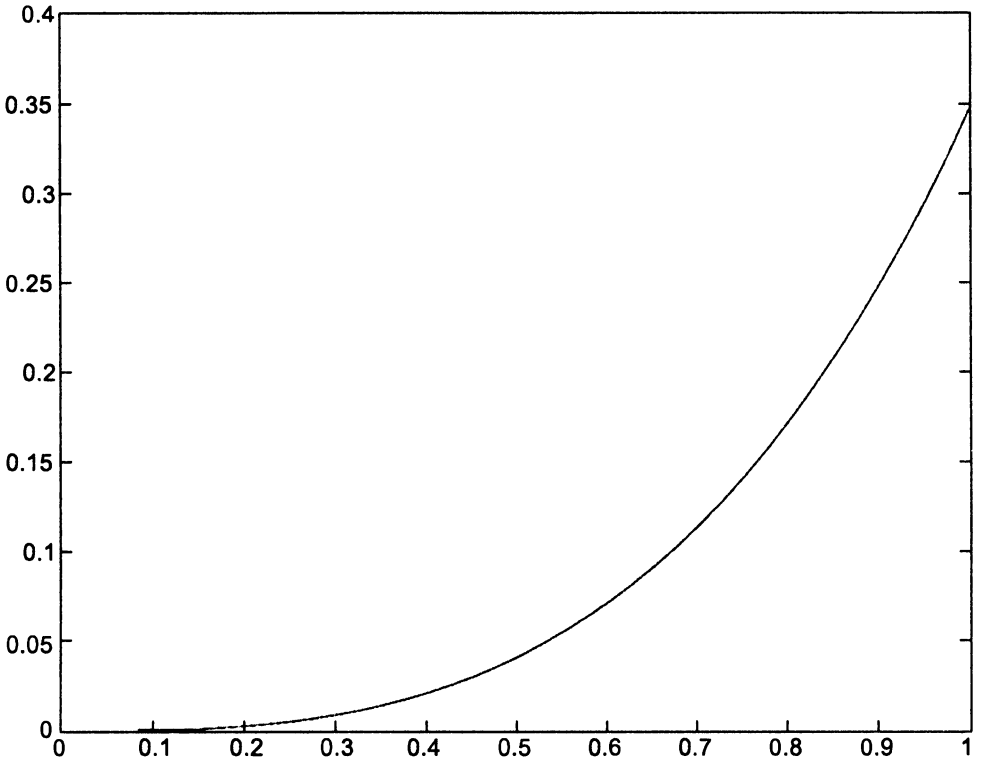

Рис. 2.1. Решение ОДУ $y' = y^2 + t^2$ с начальным условием $y(0) = 0$.

можно избежать утомительного и часто приводящего к возникновению многочисленных ошибок точного указания типов данных, что является обязательным для функций, определяемых в предварительно скомпилированных модулях. Это достигается путем широкого использования опциональных аргументов, значения которых задаются по умолчанию.

Среда программирования MATLAB позволяет осуществлять формирование и вывод массивов выходных данных, размер которых не может быть известен заранее, а также содержит средства для их графического вывода. Стандартная численная процедура для выполнения научных расчетов должна обеспечивать вывод решения во всех заданных пользователем точках сетки t_n. Разумеется, при этом могут возникнуть проблемы, связанные с выделением компьютерной памяти, достаточной для хранения соответствующего объема данных, но в любом случае значения решения в указанных точках должны быть получены. Одним из путей обеспечения этого требования в MATLAB является предусмотренная в численных процедурах решения ЗНУ встроенная возможность переопределять аргумент, задающий интервал интегрирования. Если в качестве этого интервала вы определили массив `[t0 t1 ... tf]` с более чем двумя элементами, численная процедура рассматривает этот массив как инструкцию вывести аппроксимации значений решения $y(t0), y(t1), \ldots, y(tf)$. При таком вызове численной процедуры выходной

массив `t` будет совпадать с соответствующим входным массивом. Возможность произвольного выбора числа и расположения точек временной сетки не требует существенного увеличения вычислительных затрат и одновременно имеет важное практическое значение. Выходные точки должны быть упорядочены, т. е. либо $a = t0 < t1 < \ldots < tf = b$, либо $a = t0 > t1 > \ldots > tf = b$. В качестве конкретного примера рассмотрим предыдущую ЗНУ и предположим, что необходимо построить таблицу значений решения для десяти равноудаленных точек на интервале $[0, 1]$. Это может быть сделано с использованием следующей программы

```
tout = linspace(0,1,10);
[t,y] = ode45(@f,tout,0);
```

Существует другой способ представления выходных данных численных процедур решения ЗНУ, имеющий некоторые преимущества по сравнению с рассмотренным в предыдущем абзаце. Выходные данные могут быть сформированы в виде структуры, которой может быть назначено определенное имя, например `sol`. Тогда соответствующая строка в тексте предыдущего примера должна иметь следующий вид

```
sol = ode45(@f,[0,1],0);
```

При таком вызове численной процедуры массивы выходных данных `t` и `y` становятся полями структуры `sol`. В частности, массив точек сетки `t` может быть получен как поле `sol.x`, а массив значений решения в этих точках — как поле `sol.y`. Однако эти синтаксические конструкции являются не единственным преимуществом использования структур. В действительности, численные процедуры позволяют получить также непрерывное решение $S(t)$, которое можно вычислить во всех точках интервала $[a, b]$ и для этого в указанную структуру `sol` записывается также некоторая дополнительная информация, необходимая для вычисления $S(t)$. Это реализуется с использованием вспомогательной функции `deval`, имеющей два аргумента: структуру-решение и массив точек, в которых требуется аппроксимировать решение. Таким образом, вместо вычисления с использованием численной процедуры решения в точках, заданных `tout = linspace(0,1,10)`, можно получить структуру-решение `sol` и вычислить решение в этих точках с использованием команды

```
Stout = deval(sol,tout);
```

При таком способе получения выходных данных решение ЗНУ выполняется только один раз. После этого с использованием структуры-решения можно выполнить дополнительную обработку значений решения, их графический вывод и другие действия. Таким образом, структура-решение позволяет написать функцию, определяющую решение рассматриваемой ЗНУ. Для этого в функции `deval` предусмотрен обратный порядок аргументов

```
Stout = deval(tout,sol);
```

Рассмотренная форма представления выходных данных является опциональной для численных процедур решения ЗНУ, но для численных процедур решения ЗГУ и ДУЗА эта форма вывода является единственной.

Интегрирование некоторых ОДУ не может быть продолжено после достижения сингулярной точки. Способ обработки подобной ситуации зависит от конкретной реализации численной процедуры. Эта трудность может быть легко преодолена, если численная процедура осуществляет вывод некоторых данных в каждой точке, прежде чем перейти к выполнению вычислений, которые должны быть сделаны в этой точке. В этом случае пользователь может изменить алгоритм решения задачи и не допустить проход решения через точку сингулярности. Однако подобный подход неэффективен. Современные численные процедуры выполняют интегрирование с максимально возможным шагом и используют непрерывное обобщение для аппроксимации решения в заданных выходных точках сетки. Поэтому необходимо обеспечить возможность сделать по крайней мере один шаг, выходящий за указанные пользователем пределы. Итак, возникает вопрос: что следует предпринять, если численная процедура не позволяет «пройти» некоторую точку? В большинстве стандартных процедур для научных расчетов предусмотрена опция, позволяющая пользователю информировать численную процедуру о возникновении подобной необычной ситуации. В отличие от этих процедур, численные процедуры MATLAB реализованы таким образом, что они позволяют решать задачи, заданные только на некотором заранее определенном интервале. Эти процедуры не выполняют интегрирование и не вычисляют ОДУ в точках за пределами этого интервала. Таким образом, в рассматриваемом примере для используемой численной процедуры не имеет значения определено ли соответствующее ОДУ при $t > 1$.

ПРИМЕР 2.3.2

В примере 2.3.1 рассматривалась сравнительно редкая ситуация, когда ЗНУ определялась путем набора соответствующего текста в командной строке. Программа `ch2tx1.m` иллюстрирует более широко используемый способ вызова численных процедур в среде MATLAB. Эта программа позволяет решить на интервале $0 \leqslant t \leqslant 8 \cdot 10^5$ систему ОДУ

$$\begin{aligned} x_1' &= -k_1 x_1 + k_2 y, \\ x_2' &= -k_4 x_2 + k_3 y, \\ y' &= k_1 x_1 + k_4 x_2 - (k_1 + k_3) y \end{aligned}$$

с начальными условиями

$$x_1(0) = 0,\ x_2(0) = 1,\ y(0) = 0.$$

В примере использовались численные значения коэффициентов

$$k_1 = 8.4303270 \cdot 10^{-10},\ k_2 = 2.9002673 \cdot 10^{11},$$

$$k_3 = 2.4603642 \cdot 10^{10},\ k_4 = 8.7600580 \cdot 10^{-6}.$$

Текст основной программы можно представить в виде функции `odes`, выходным параметром которой является вектор-столбец, содержащий вычисленное значение

правой части системы ОДУ. В рассматриваемом примере этот вектор определяется путем инициализации `dydt` нулями. Аппроксимации значений решения $y_i(t_n)$, $n = 1, 2, \ldots$ доступны через выходные значения `y(:,i)`. В программе эти выходные данные используются при построении графиков решения $x_1(t)$ и $x_2(t)$, представленных в линейном по t масштабе, и отдельного графика для $y(t)$, представленного в логарифмическом масштабе по t.

В этом примере рассматривается задача о протонном переносе (см. параграф 1.4; рисунки 1.6 и 1.7 соответствуют двум упомянутым выше графикам решения). Мы отмечали, что принятое по умолчанию допустимое значение абсолютной ошибки 10^{-6}, используемое при вычислении всех компонент решения является неприемлемым при вычислении $y(t)$, т. к. из рисунка 1.7 видно, что абсолютная величина этой компоненты меньше $3 \cdot 10^{-17}$. Всем опциональным аргументам численных процедур MATLAB могут быть присвоены определенные значения с использованием дополнительной функции `odeset`, а также специальных *ключевых слов*. Краткую справку по опциональным аргументам и о самой функции `odeset` можно получить, набрав в командной строке `help odest`. На практике наиболее часто изменяемыми параметрами являются допустимые значения ошибки. Для рассматриваемой задачи наиболее подходящим значением допустимой абсолютной ошибки является 10^{-20}. Если допустимое значение абсолютной ошибки задается в виде скалярной величины, с учетом этого значения вычисляются все компоненты решения. Если эта величина является векторной, то компоненты решения будут вычисляться с точностью, заданной соответствующей компонентой вектора допустимого значения абсолютной ошибки. Входные значения численной процедуры, которые оказались не определенными пользователем, принимают перед началом ее работы значения, установленные для соответствующих величин по умолчанию. В рассматриваемом примере использовалось установленное по умолчанию допустимое значение относительной ошибки 10^{-3} (всегда скаляр), но если мы хотим изменить это значение, например, на 10^{-4}, следует использовать следующую команду

```
options = odeset('AbsTol',1e-20,'RelTol',1e-4)
```

Опциональные аргументы могут быть представлены в списке аргументов численной процедуры в любом порядке; ключевые слова используются без учета регистра клавиатуры. С использованием функции `odeset` формируется структура опций, которая впоследствии может использоваться в качестве опционального входного аргумента численной процедуры. Мы выбрали идентификатор `options`, поскольку это имя представляется естественным для структуры опций, но вы можете использовать любой другой идентификатор.

```
function ch2ex1
% x_1 = y(1), x_2 = y(2), y = y(3)

options = odeset('AbsTol',1e-20);
[t,y] = ode15s(@odes,[0 8e5],[0; 1; 0],options);
plot(t,y(:,1:2))
figure
semilogx(t,y(:,3))
```

```
%================================================
function dydt = odes(t,y)
k = [8.4303270e-10 2.9002673e+11 2.4603642e+10 8.7600580e-06];
dydt = zeros(3,1);
dydt(1) = -k(1)*y(1) + k(2)*y(3);
dydt(2) = -k(4)*y(2) + k(3)*y(3);
dydt(3) = k(1)*y(1) + k(4)*y(2) - (k(2) + k(3))*y(3);
```

Для решения этой задачи мы выбрали основанную на МДН численную процедуру `ode15s`, поскольку эта ЗНУ является жесткой. С точки зрения синтаксиса в простейших случаях все численные процедуры для решения ЗНУ взаимозаменяемы. Например, для того, чтобы использовать метод Рунге–Кутта, реализованный в численной процедуре `ode45`, следует соответствующим образом изменить имя функции, вызываемой в программе `ch2ex1.m`. Читателю предлагается сделать это в упражнении 2.18 и убедиться в том, что для решения жестких задач действительно может потребоваться использование специальных численных процедур.

ПРИМЕР 2.3.3

По умолчанию выходные данные численных процедур интегрирования ЗНУ содержат все точки, в которых вычислены аппроксимации решения. Для модификации формы вывода выходных данных можно написать специальную *функцию вывода*, которая будет вызываться в используемой численной процедуре решения ЗНУ каждый раз, когда осуществляется вывод результатов вычислений на очередном шаге интегрирования. В системе MATLAB реализованы несколько стандартных функций вывода. Если пользователь не задал выходные аргументы, то по умолчанию в качестве функции вывода будет использована функция вывода `odeplot`, которая осуществляет графический вывод всех компонент решения по мере их получения в процессе интегрирования ЗНУ на промежутке интегрирования. В упражнениях 2.15 и 2.18 рассмотрены несколько иллюстративных примеров. Часто бывает, что нет необходимости в выводе всех компонент решения. Выбор выводимых в процессе вычислений компонент может быть осуществлен путем задания опции `OutputSel`. Существуют и другие стандартные функции вывода, которые предназначены, например, для вывода фазовых портретов исследуемой системы. Пример подобной программы рассмотрен в упражнении 2.21.

Наряду с рассмотренными в этой книге примерами в качестве иллюстрации использования численных процедур решения ЗНУ можно использовать программу `dfs.m`, которая представляет собой программную реализацию стандартной задачи численного решения скалярного ОДУ $y' = f(t, y)$ с графическим выводом полученного решения. В упражнении 1.4 обсуждались различные аспекты использования этой программы, но здесь мы воспользуемся ею для того, чтобы показать на ее примере как можно написать свою функцию вывода. В некоторых современных численных процедурах предусмотрена возможность задания минимального шага интегрирования `hmin`, а также возможность прерывания выполнения вычислений, если эта величина становится меньше этого минимального значения. В численных процедурах MATLAB минимальное значение шага интегрирования не может быть установлено, но в программе `dfs.m` эта функциональность реализована с исполь-

зованием специальной функции вывода. Соответствующий отрывок текста этой программы приведен ниже

```
function dfs(fun,window,npts)
.
.
.
hmin = 1e-4*(wR - wL);
.
.
.
options = odeset('Events',@events,'OutputFcn',@outfcn,...
.
.
.
[t,y] = ode45(@F,[t0,wR],y0,options);
plot(t,y,'r')
[t,y] = ode45(@F,[t0,wL],y0,options);
plot(t,y,'r')
.
.
.
function status = outfcn(t,y,flag)
global hmin
persistent previoust
status = 0;
switch flag
case 'init'
previoust = t(1);
case ''
h = t(end) - previoust;
previoust = t(end);
status = (abs(h) <= hmin);
case 'done'
clear previoust
end
```

С помощью опции `OutputFcn` задается имя функции, которая будет использоваться в соответствующей численной процедуре в качестве функции вывода, в рассматриваемом случае `outfcn`. Если задана функция вывода, численная процедура вызывает ее на каждом шаге с аргументами `t,y,flag`, где `t` и `y` — значения независимой и зависимой переменных и строка `flag` (флаг) определяет различные опции вызова этой функции. Выходной переменной функции вывода является `status`. Если после анализа входных переменных принимается решение о продолжении выполнения процедуры интегрирования, выходной переменной должно быть присвоено значение 0 (false); если решено прервать выполнение программы, выходному значению должно быть присвоено значение 1 (true). Численная процедура вызывает функцию вывода в первый раз с флагом 'init', что позволяет пользователю выполнить инициализацию этой функции. Например, если вы намерены

записывать выходные данные в файл, во время инициализации вы можете открыть этот файл. Кроме того, при первом вызове функции вывода в аргументе `t` записан интервал интегрирования $[a, b]$, а в аргументе `y` — начальное значение $y(a)$. В рассматриваемом примере переменная `previoust` инициализируется начальным значением, равным значению независимой переменной `t(1)`. Эта переменная декларирована как `persistent` и поэтому ее значение будет сохраняться при выходе из функции вывода и восстанавливаться при очередном ее вызове. После выполнения очередного шага интегрирования численная процедура вызывает функцию вывода с флагом `' '`. В общем случае аргумент `t` содержит значение независимой переменной в конце шага, а аргумент `y` — соответствующее значение аппроксимации решения. Однако в случае использования численной процедуры `ode45`, как в рассматриваемом примере, ситуация несколько усложняется. Численные процедуры имеют опцию `Refine`, принимающую целые значения, применение которой приводит к тому, что численная процедура возвращает указанное количество аппроксимаций решения, равноразнесенных на интервале шага интегрирования. По умолчанию значение `Refine` равно 1 для всех численных процедур за исключением `ode45`, для которой оно равно 4. В функции вывода эти аппроксимации решения могут быть обработаны в соответствии с запросами пользователя. Например, значения `t` и выбранных компонент `y` можно записать в файл. Если `Refine` больше 1, аппроксимации решения будут упорядочены, что позволяет пользователю получить значение независимой переменной в конце шага `t(end)`, которое в свою очередь может использоваться, как в рассматриваемом примере, для вычисления текущего значения шага интегрирования. Если этот шаг интегрирования не превышает минимального значения, переменной `status` присваивается значение 1, что прерывает процесс интегрирования. В упражнении 2.29 читателю предлагается модифицировать функцию вывода в программе `dfs.m` так, чтобы реализовать прерывание выполнения программы при других обстоятельствах. После выполнения вычислений на последнем шаге интегрирования численная процедура вызывает функцию вывода с флагом 'done', что позволяет выполнить заключительные операции вывода, например закрыть файл, в который записывались выходные данные. В рассматриваемой программе во время этого заключительного вызова происходит инициализация переменной `previoust`.

■ УПРАЖНЕНИЕ 2.17

В главе 1 мы убедились, что численное интегрирование дифференциальных уравнений

- $y' = y^2 + 1$,
- $y' = y^2 + t$,
- $y' = y^2 + t^2$

может быть выполнено в системе MATLAB с использованием единой процедуры. Убедитесь в этом на практике, решив эти уравнения с применением, скажем, численной процедуры `ode45`. В качестве интервала интегрирования используйте $[0, 1]$, а в качестве начальных условий $y(0) = 0$. Выведите полученные решения графически.

■ УПРАЖНЕНИЕ 2.18

Модифицируйте текст программы `ch2ex1.m` так, чтобы графический вывод аппроксимации решения осуществлялся в процессе выполнения вычислений. В примере 2.3.3 мы убедились, что эта цель может быть достигнута, если не указывать выходные аргументы. При тестовых запусках своей программы вы убедитесь в том, что численная процедура `ode15s` позволяет решить задачу достаточно быстро. С другой стороны, при использовании численной процедуры `ode45` может показаться, что вычисления прекращаются непосредственно в начальной точке, однако это не так. Для того, чтобы убедиться в том, что процедура продолжает свою работу, можно нажать кнопку «Stop», что приведет к нормальному прерыванию вычислительного процесса и графическому выводу полученных результатов в масштабе, который позволит вам оценить насколько далеко продвинулся процесс интегрирования за время вашего ожидания. Это подтверждает тот факт, что `ode45` может решить рассматриваемую ЗНУ, но этот процесс может длиться очень долго.

■ УПРАЖНЕНИЕ 2.19

Во времена, когда эффективные программы для решения жестких задач еще не имели столь широкого распространения как в настоящее время, некоторые жесткие задачи решались с использованием методов сингулярных возмущений в комбинации со стандартными численными методами для нежестких задач. Применяя этот подход, Лапидус с соавторами [Lapidus et al., 1973] исследовали модель термического разложения озона

$$\frac{dx}{dt} = -x - xy + \varepsilon\kappa y,$$
$$\varepsilon\frac{dy}{dt} = x - xy - \varepsilon\kappa y,$$

где x — приведенная концентрация (reduced concentration) озона, y — приведенная концентрация кислорода, $\varepsilon = 1/98$ и $\kappa = 3$ — параметры модели. (Заметим, что $y'(t)$ умножается на малый параметр ε.) Эта ЗНУ должна быть решена на интервале $[0, 240]$ с начальными условиями $x(0) = 1$ и $y(0) = 0$. Лапидус пришел к несколько неожиданному выводу, заключающемуся в том, что методы сингулярных возмущений не позволяют получить достаточно точное решение этой ЗНУ, поскольку параметр ε недостаточно мал, т. е. задача при указанной величине этого параметра является недостаточно жесткой. В настоящее время численное решение подобных задач не представляет проблему. Решите эту ЗНУ с использованием `ode15s` и отобразите графически полученное решение $y(t)$ с использованием процедур `semilogx` и `axis([0.01 100 0 1])`. Сравните результат с рисунком 3 из работы [Lapidus et al., 1973]. Убедитесь в том, что рассматриваемая ЗНУ является не очень жесткой путем ее решения с использованием `ode45`. Для этого с использованием `odeset` установите опцию `Stats` в состояние `on`, выполните вычисления с применением указанных двух численных процедур и сравните статистику.

■ УПРАЖНЕНИЕ 2.20

Ранее отмечалось, что если рассматриваемая ЗНУ является жесткой, то при ее решении с использованием неявных методов необходимая для вычисления

соответствующих формул итерационная матрица является плохо обусловленной. Убедитесь в этом, модифицировав программу `ch2ex1.m` так, чтобы в процессе интегрирования для каждой итерационной матрицы выводилось ее число обусловленности. Для этого скопируйте файл `ode15s.m` в вашу рабочую папку и переименуйте его в `mode15s.m`. Исходный программный код процедуры `ode15s` может быть найден в папке `/toolbox/matlab/funfun/`, расположенной в папке, в которую была установлена система MATLAB. Кроме того, вам необходимо скопировать дополнительные функции, которые вызываются в `ode15s`: `ntrp15s.m`, `odearguments.m`, `odeevents.m`, `odejacobian.m` и `odemass.m`. Соответствующие файлы могут быть найдены в папке `/toolbox/matlab/funfun/private/`. Для выполнения модификации, необходимой для вычисления числа обусловленности итерационной матрицы, найдите в `mode15s.m` строку

```
[L,U] = lu(Miter);
```

в которой происходит вызов функции `lu`, предназначенной для вычисления LU-декомпозиции матрицы. Вызов этой функции встречается в тексте программы дважды: первый раз при инициализации численной процедуры и второй раз в основном цикле программы. В первом случае вы должны после соответствующей строки ввести следующие команды

```
global tcond cond
tcond = t;
cond = condest(Miter);
```

С использованием этих команд инициализируется пара выходных массивов и вычисляется оценка числа обусловленности итерационной матрицы в терминах 1-нормы путем вызова функции `condest`. Заметим, что при таком определении переменных, их значения оказываются доступными вне численной процедуры. После второго вызова функции `lu` следует ввести команды

```
tcond = [tcond t];
cond = [cond condest(Miter)];
```

которые позволяют после каждого выполнения LU-разложения итерационной матрицы увеличить область памяти, в которой хранятся выходные массивы, и записать в эти массивы информацию, касающуюся числа обусловленности очередной итерационной матрицы. После выполнения всех этих модификаций процедуры `mode15s` можно легко осуществлять мониторинг текущего значения числа обусловленности итерационной матрицы в процессе решения ЗНУ с использованием программы `ch2ex1.m`, которая также должна быть модифицирована следующим образом: для получения доступа к определенным выше глобальным переменным добавляем строку

```
global tcond cond
```

и выполняем вычисления с графическим выводом соответствующих результатов

```
figure
loglog(tcond,cond,'*-')
```

Анализ результатов показывает, что на первом участке интервала интегрирования число обусловленности итерационных матриц сравнимо с наименьшим возможным значением 1. На втором участке интервала интегрирования число обусловленности монотонно возрастает до значения, сравнимого с максимально возможным значением `1/eps`. Число обусловленности несколько уменьшается на последнем шаге интегрирования. Объясните подобное поведение с учетом графиков компонент решения, полученных с использованием программы `ch2ex1.m` и графика значений шага интегрирования, который может быть получен с использованием команд

```
figure
loglog(t(1:end-1),diff(t))
```

■ УПРАЖНЕНИЕ 2.21

В MATLAB реализованы две стандартный функции вывода `odephas2` и `odephas3`, которые позволяют получить графическое изображение фазовых портретов решений, полученных с использованием численных процедур интегрирования ЗНУ. Решения системы ОДУ

$$y_1' = -y_2 - \frac{y_1 y_3}{r},$$
$$y_2' = y_1 - \frac{y_2 y_3}{r},$$
$$y_3' = \frac{y_1}{r},$$

где

$$r = \sqrt{y_1^2 + y_2^2},$$

лежат на торе в фазовом пространстве. Используя `ode45`, найдите решение этой системы на интервале $0 \leqslant t \leqslant 10$ с начальными данными

$$(y_1(0). y_2(0), y_3(0)) = (3, 0, 0).$$

В процессе интегрирования постройте графическое представление полученного решения в трехмерном фазовом пространстве, задав при вызове численной процедуры `@odephas3` соответствующую функцию вывода `outputFcn`. Процедуры отображения графиков соединяют последовательные точки кривых прямыми линиями, поэтому при достаточно больших шагах интегрирования график решения может иметь вид ломаной линии. Этот эффект особенно отчетливо проявляется при рисовании фазовых портретов. Если возникает подобная ситуация, можно воспользоваться опцией `Refine`, которая позволяет вычислить и вывести равноразнесенные на интервале текущего шага интегрирования дополнительные промежуточные аппроксимации решения. Это осуществляется с использованием соответствующего непрерывного обобщения используемого численного метода. Для ознакомления с использованием этой опции постройте два графика. В первом случае значение `Refine` должно быть равно 1, т. е. на каждом шаге будет выводиться только одна точка. Для процедуры `ode45` установленное по умолчанию значение `Refine` равно 4, но при построении второго графика установите значение `Refine` равным 10.

При этом удобно воспользоваться функцией odeset, которая позволяет устанавливать и изменять значения опций численных процедур интегрирования. Перед построением первого графика следует набрать команду

```
options = odeset('OutputFcn',@odephas3,'Refine',1);
```

а перед построении второго значение Refine может быть изменено командой

```
options = odeset(options,'Refine',10);
```

■ УПРАЖНЕНИЕ 2.22

В работе [Raghothama & Narayanan, 2002] рассматривается вопрос о параметрическом возмущении следующего ОДУ

$$x''(t) = -2\zeta x'(t) - x - 1.5x^2(t) - 0.5x^3(t) + f_2\cos(\Omega_2 t) + f_3\cos(\Omega_3 t),$$

где $f_2 = 0.05$, $\Omega_2 = 1$, $\Omega_3 = 2$, $\zeta = 0.1$ и f_3 — бифуркационный параметр. Авторы показали, что при изменении f_3 изменяются качественные характеристики фазовых кривых $(x(t), x'(t))$. Используя ode45, найдите решение этой ЗНУ для четырех случаев, соответствующих графикам, изображенным на рисунке 9 в работе [Raghothama & Narayanan, 2002]. Интегрирование выполните при RelTol = 1e-8 и AbsTol = 1e-8 на интервале от $t = 0$ до $t = 500$. Поведение решений в предельном режиме может быть получено, если выполнить интегрирование при $400 \leqslant t \leqslant 500$. Если интегрирование было выполнено с использованием команды вида [t, x] = ode45 ..., указанная цель может быть достигнута путем нахождения на интервале интегрирования соответствующих точек сетки командой ndx = find(t >= 400) с последующим графическим выводом результатов с использованием команды plot(x(ndx,1),x(ndx,2)). Полученные в результате интегрирования ОДУ фазовые портреты должны соответствовать периодическому движению с периодом 1, 2, 4 и 8, соответственно.

- $x(0) = 0, \quad x'(0) = -0.60, \quad f_3 = 0.50,$
- $x(0) = 0, \quad x'(0) = -0.80, \quad f_3 = 0.92,$
- $x(0) = 0, \quad x'(0) = -0.59, \quad f_3 = 0.99,$
- $x(0) = 0, \quad x'(0) = -0.80, \quad f_3 = 1.005.$

Далее в своей работе авторы показали, что решения исследуемого уравнения могут демонстрировать хаотическое поведение. Найдите решение этой ЗНУ при $x(0) = 0$, $x'(0) = 0$ и $f_3 = 1.35$. Соответствующая кривая в фазовом пространстве должна быть похожа на перчатку кетчера.[1]

2.3.1. Локализация событий

Ранее уже упоминалась программа dfs.m, которая может служить примером численного решения скалярного ОДУ $y' = f(t, y)$ с последующим графическим выводом результатов вычислений. В упражнении уже обсуждалось использование этой программы, и здесь мы рассмотрим некоторые дополнительные детали. Решение отображается на экране компьютера в специальном окне, параметры которого определяются массивом [wL, wR, wB, wT]. Если в этом окне кликнуть мышью в

[1] Прим. перев. — catcher — кетчер, игрок в бейсболе, принимающий подачу.

некоторой точке (t_0, y_0), программа вычисляет соответствующее решение $y(t)$ с начальным значением $y(t_0) = y_0$ и осуществляет графический вывод этого решения при $wL \leqslant t \leqslant wR$ и $wB \leqslant y \leqslant wT$. Это осуществляется в два этапа. На первом этапе происходит вызов функции `ode45` для интервала интегрирования $[t_0, wR]$, в результате чего вычисляется решение $y(t)$ на промежутке времени от начального момента до момента времени, соответствующего правому краю окна. После этого происходит вызов численной процедуры для интервала интегрирования $[t_0, wL]$ и вычисляется решение $y(t)$ на промежутке времени от начального момента до момента времени, соответствующего левому краю окна. Но что произойдет, если график решения выйдет за пределы верхнего или нижнего края окна? Необходимо предусмотреть возможность прерывания вычислительного процесса, если возникла подобная ситуация, т. е. когда при некотором t^* выполнено $y(t^*) = wB$ или $y(t^*) = wT$.

При получении аппроксимации решения ЗНУ может возникнуть необходимость в нахождении (локализации) точек t^*, в которых значения определенных *функций обработки событий* (event functions)

$$g_1(t, y(t)), g_2(t, y(t)), \ldots, g_k(t, y(t))$$

становятся равными нулю. Определение этих точек называется процедурой *локализации событий* (event location). Иногда бывает необходимо просто знать значение решения $y(t^*)$ в момент t^* и подобная ситуация может рассматриваться как простейшее событие, требующее соответствующей обработки. В других случаях может возникнуть необходимость прервать вычислительный процесс в момент t^* и возможно начать решение новой ЗНУ с другим ОДУ и новыми начальными данными, которые определяются в зависимости от значений t^* и $y(t^*)$. Иногда принятие решения может зависеть от характера изменения значений функции обработки событий, т. е. от того убывают ли эти значения или возрастают в момент возникновения события. В программе `dfs.m` используются две функции обработки событий (на основе анализа значений $wB - y$ и $wT - y$); обе эти функции приводят в соответствующих случаях к прерыванию выполнения программы, причем в момент наступления события анализ характера изменения значений этих функций не выполняется. Во всех численных процедурах MATLAB, предназначенных для решения ЗНУ, реализован мощный механизм локализации событий. Большинство специальных языков программирования и численных процедур для решения ЗНУ, предназначенных для проведения научных расчетов, также снабжены соответствующими средствами для локализации событий. Следует иметь в виду, что процедуры локализации событий могут быть реализованы некорректно и некоторые используемые алгоритмы часто бывают очень примитивны. Сложность задачи написания адекватных функций обработки событий на практике иногда просто игнорируется разработчиком, и поэтому в этом параграфе мы попытаемся привить читателю чувство осторожности, которое должно уберечь его от стандартных ошибок при использовании аппарата обработки событий, реализованного в MATLAB. Более детальное обсуждение этого вопроса может быть найдено в работах [Shampine, Gladwell & Brankin, 1991] и [Shampine & Thompson, 2000].

В большинстве программ, использующих локализацию событий, во время выполнения шага интегрирования функция обработки событий осуществляет

мониторинг знака определенной величины. Если, скажем, $g_i(t_n, y_n)$ и $g_i(t_{n+1}, y_{n+1})$ имеют различные знаки, то численная процедура использует стандартные численные методы для нахождения корня алгебраического уравнения $g_i(t, y(t)) = 0$ на интервале $[t_n, t_{n+1}]$. Для нахождения корня необходимо вычислить функцию обработки событий g_i в нескольких точках t. Очевидно, что непрерывное обобщение используемого численного метода интегрирования приобретает в рассматриваемой ситуации особенное значение, т. к. для вычисления $g_i(t, y(t))$ необходимо знать аппроксимации $y(t)$ для всех t на указанном интервале. Без использования интерполяции вычисление функций g_i во все моменты t на интервале $[t_n, t_{n+1}]$ пришлось бы выполнять с использованием аппроксимаций $y(t)$, полученных численной процедурой интегрирования ОДУ на промежутке между t_n и t. Понятно, что более эффективным путем решения задачи является использование интерполяционного полинома $S(t)$ для нахождения значений $y(t)$, необходимая точность которых обеспечивается на всем интервале $[t_n, t_{n+1}]$. В этом случае задача локализации события сводится к вычислению нулей уравнения $g_i(t, S(t)) = 0$. Часто используется несколько функций обработки событий. Все эти функции должны анализироваться одновременно, т. к. возможна ситуация, когда в процессе нахождения корня одной функции обработки события выясняется, что другая функция обработки событий также имеет корень, значение которого ближе к t_n. Всегда следует выявлять такое событие, момент возникновения которого наиболее близок к t_n, т. к. именно в этот момент может произойти переопределение ОДУ.

Приведенные выше предварительные замечания о задаче локализации событий должны способствовать более внимательному отношению к ее решению. Рассмотренный подход не позволяет выявить факт изменения знака, если этот знак изменился четное количество раз (в этом случае говорят, что функция обработки события имеет нуль четного порядка (even-order zero)), т. к. в этом случае знак фактически не меняется. Кроме того, возможна также ситуация, когда не удается выявить нечетное количество изменений знака (нули нечетного порядка (odd-order zero)). Это может произойти при дроблении первоначального шага интегрирования в целях выявления деталей изменения решения. Указанное дробление шага может не привести к изменению финального значения функции обработки ошибок в конце этого шага, т. к. программа может выполнить несколько уменьшенных шагов и выявить несколько возникновений соответствующего события, но на заключительном уменьшенном шаге, соответствующем окончанию первоначального шага, может оказаться, что знак фактически остался неизменным. Аналогичная проблема часто возникает в приложениях при вычислении корня функции обработки событий в первый раз: вычисленное значение должно быть единственным и наиболее близким к t_n, но в общем случае у нас нет уверенности в том, что полученное значение удовлетворяет этим свойствам. Некоторые используемые на практике функции обработки событий не являются гладкими, что усложняет вычисление их корней. В действительности разрывные функции обработки событий используются не так редко, как могло бы показаться. Так же как и в случае вычисления корней алгебраического уравнения, корень функции обработки событий может быть плохо обусловленным, т. е. значение t^* плохо определяется имеющимися значениями функции обработки событий. При нахождении корней обычно предполагается, что функция может быть вычислена очень точно, но в

рассматриваемой ситуации для вычислений используются значения $y(t)$, известные лишь с определенной точностью. Таким образом, фундаментальная проблема заключается в том, чтобы определить точность локализации события t^* с учетом того, что значения функции $y(t)$ вычисляются с известной, но не абсолютной точностью. Этот вопрос должен рассматриваться не только на этапе постановки задачи, но и при реализации численной процедуры вычисления корней, когда следует определить необходимую точность вычисления t^*. В некоторых численных процедурах допустимые ошибки при вычислении функций обработки событий задаются пользователем. Однако определить это значение осмысленно обычно бывает трудно и поэтому часто (например в MATLAB) используется другой подход, заключающийся в том, что локализация события выполняется с той точностью, которая может быть достигнута на используемом компьютере. Для того, чтобы этот подход имел практическое значение, необходимо, чтобы алгоритм вычисления корней позволял быстро получить требуемый результат в стандартном случае, когда функция обработки событий является гладкой. Этот алгоритм должен быть также достаточно быстр, когда эта функция не является гладкой. В статье [Moler, 1997] обсуждается используемый в MATLAB алгоритм, в котором применяется робастный и достаточно быстрый метод деления пополам, пригодный для гладких функций обработки событий и допускающий векторизацию.

Рассмотренные ниже примеры показывают, что при использовании численных процедур MATLAB реализовать локализацию событий нетрудно. Действительно, единственная трудность, которую пользователь должен преодолеть — это определиться с выбором локализуемого события и действий, которые должны быть выполнены в численной процедуре в случае, когда это событие возникает. Однако проблемы, связанные с локализацией события, часто бывают нетривиальны вследствие того, что эту локализацию приходится выполнять уже после того, как событие произошло. В примере 2.3.4 указанная проблема решается алгоритмически. Аналогичные ситуации рассматриваются в упражнениях 2.25 и 2.26. В примере 2.3.5 рассматривается более сложная ситуация: после локализации события формулируется и решается новая ЗНУ. Упражнения 2.27 и 2.28 представляют собой различные варианты этого примера. Пример и пара упражнений, рассмотренные в последующих параграфах, иллюстрируют случаи решения проблемы локализации события при нестандартной постановке ЗНУ. Например, система ОДУ в примере 2.3.7 представляется с использованием матрицы весовых коэффициентов. Упражнение 2.32 является в некотором роде обобщением этого примера. Система ОДУ в упражнении 2.37 является сингулярной в начальной точке.

Для того, чтобы убедится в том, что задача локализации события на практике может оказаться достаточно простой, обсудим соответствующую часть программы `dfs.m`. Функция `events` реализует две функции обработки событий, используемые в численной процедуре решения задачи при принятии решения в соответствующих ситуациях. Указатель на эту функцию передается в процедуру с использованием опции `Events`. Входными значениями функции `events` являются `t`, `y`, и значения функций обработки событий записываются в выходную переменную `value`. Компоненты выходного вектора `isterminal` содержат информацию для численной процедуры о том, что соответствующее событие является терминальным (т. е. требует прерывания вычислений). В рассматриваемом случае оба события являются

терминальными. Компоненты выходного вектора direction содержат информацию для численной процедуры о том, что при анализе значения соответствующей функции обработки события следует учитывать факт увеличения или уменьшения этого значения в момент наступления события. При возникновения локализуемого события численная процедура имеет некоторые дополнительные выходные аргументы, но поскольку в рассматриваемом случае нам необходимо лишь прервать процесс интегрирования, мы не требуем выводить эти аргументы.

```
function dfs(fun,window,npts)
.
.
.
options = odeset('Events',@events,...
.
.
.
[t,y] = ode45(@F,[t0,wR],y0,options);
plot(t,y,'r')
[t,y] = ode45(@F,[t0,wL],y0,options);
plot(t,y,'r')
.
.
.

function [value,isterminal,direction] = events(t,y)
global wB wT
value = [wB; wT] - y;
isterminal = [1; 1];
direction = [0; 0];
```

Локализация событий в упражнении 2.23 может быть реализована аналогичным образом, но в этом случае полезно использовать дополнительные выходные переменные численной процедуры. Рассмотренный пример может рассматриваться лишь в качестве общей схемы использования в MATLAB аппарата локализации событий. Дополнительные детали этой проблемы будут выявлены в ходе рассмотрения последующих примеров.

ПРИМЕР 2.3.4

Отображение Пуанкаре является важным средством для изучения качественных характеристик динамических систем. Графики этих отображений часто представляют собой вычисленные в равноразнесенные моменты времени точки решения. Соответствующие значения могут быть легко получены путем задания массива этих моментов времени tspan. Однако в некоторых случаях бывает необходимо, чтобы эти моменты удовлетворяли требованию равенства нулю некоторой линейной комбинации значений решения и тогда для вычисления этих моментов времени используется соответствующая функция обработки события. В работе [Koçak, 1989. Уроки 13 и 14.] эта задача решена с использованием программы PHASER. В нашем примере мы выполним численное интегрирование соответствующей системы из четырех ОДУ с переменными $x_1(t), x_2(t), x_3(t)$ и $x_4(t)$, описывающей движение

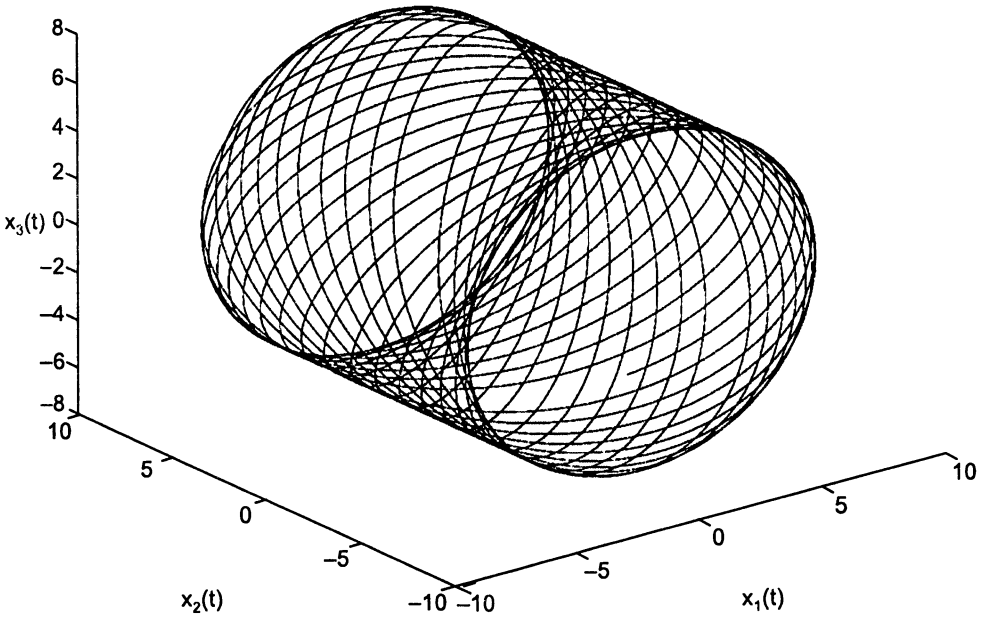

Рис. 2.2. Фигура Лиссажу для пары гармонических осцилляторов.

пары гармонических осцилляторов. Уравнения этой системы и начальные условия задаются в программе `ch2ex2.m`. Графики $x_1(t), x_2(t)$ и $x_3(t)$ показывают, что соответствующие траектории лежат на ограниченном цилиндре (см. рис. 2.2). После выполнения вычислений вы можете кликнуть мышью кнопку `Rotate 3D` и повернуть график так, чтобы выбрать наиболее удобную для восприятия проекцию решения. С использованием программы `PHASER` могут быть найдены и отображены на графике точки, удовлетворяющие уравнению

$$Ax_1(t) + Bx_2(t) + Cx_3(t) + D = 0.$$

В работе [Koçak, 1989] получены два подобных отображения Пуанкаре. Первое отображение Пуанкаре соответствует паре координат $(x_1(t^*), x_3(t^*))$, вычисленных при значениях t^* и удовлетворяющих $x_2(t^*) = 0$. Второе отображение Пуанкаре соответствует паре координат $(x_1(t^*), x_2(t^*))$, вычисленных при значениях t^* и удовлетворяющих $x_3(t^*) = 0$. Эти решения будут получены нами с использованием функций обработки события $g_1 = x_2$ и $g_2 = x_3$, вычисление которых будет выполняться в процедуре `events`. В число параметров численной процедуры решения рассматриваемой ЗНУ должна входить опция `Events` с соответствующим указателем на эту функцию обработки событий. Входными аргументами функции обработки событий являются `t`, `x` и в качестве выходной переменной используется вектор-столбец значений функций обработки отдельного события $g_1(t, x)$ и $g_2(t, x)$. Программная реализация функции обработки событий `events` должна предполагать возможность вывода некоторой информации, определяющей алгоритм дальнейших вычислений. В некоторых случаях может возникнуть необходимость в

прекращении процесса дальнейшего интегрирования при возникновении определенного события. Для этого используется второй выходной аргумент функции обработки событий `isterminal`, являющийся вектор-столбцом, компоненты которого могут принимать два значения: 0 и 1. Если необходимо прервать процесс интегрирования при $g_k(t,x) = 0$, соответствующей компоненте вектора `isterminal(k)` присваивается значение 1, в противном случае — 0. Задание начальных значений требует особого внимания, т. к. иногда событие, являющееся в соответствии с нашим определением терминальным, может произойти именно в начальный момент времени (см., например, упражнения 2.3.5 и 2.24). Поскольку подобная ситуация случается нередко, любое событие в начальный момент времени рассматривается численными процедурами как нетерминальное. Иногда бывает важно учитывать факт увеличения или уменьшения значений функции обработки события в момент возникновения события. Для этого используется третий аргумент функции обработки событий `direction`, являющийся вектор-столбцом. Если следует обрабатывать только те события, при возникновении которых (т. е. при $g_k(t,x) = 0$) функция обработки события g_k является убывающей, соответствующей компоненте этого вектора `direction(k)` следует присвоить значение -1. Аналогично, если следует обрабатывать только те события, при возникновении которых функция обработки события g_k является возрастающей, компоненте `direction(k)` следует присвоить значение $+1$. Если следует обрабатывать все события, связанные с возрастанием или убыванием значений функции обработки события, компоненте `direction(k)` следует присвоить значение 0. В рассматриваемом примере все события являются нетерминальными и следует обрабатывать все события, связанные с возрастанием или убыванием значений функции обработки события.

При использовании аппарата локализации событий вектор выходных переменных численной процедуры решения ЗНУ включает дополнительные компоненты: `[t,x,te,xe,ie] = ode45...` Массивы `te` и `xe` содержат значения независимой переменной, при которых произошло событие, и значения решения в эти моменты времени соответственно. Компонентами массива `ie` являются целые числа, соответствующие последовательному номеру функции обработки события, оказавшейся равной нулю в момент возникновения события; значение этого момента времени записано в соответствующем компоненте вектора `te`. (Если вы предпочитаете использовать выходные данные в виде единой структуры `sol`, вся эта выходная информация о возникновении событий записывается в соответствующие поля этой структуры `sol.xe`, `sol.ye` и `sol.ie`.) Если в процессе интегрирования не произошло ни одного события, все эти массивы являются пустыми, поэтому для установления факта возникновения событий достаточно проверить значение функции `isempty(ie)`. Одной из проблем, возникающей при локализации событий, является то, что при наличии нескольких функций обработки события нам неизвестен порядок возникновения различных событий. Для преодоления этой трудности удобно воспользоваться функцией `find` (см. текст программы). Например, команда

```
event1 = find(ie == 1);
```

позволяет получить вектор `event1`, содержащий индексы, соответствующие случаям обработки первой функции обработки события. Эти индексы могут быть использованы для получения соответствующих значений решения `xe` и построения

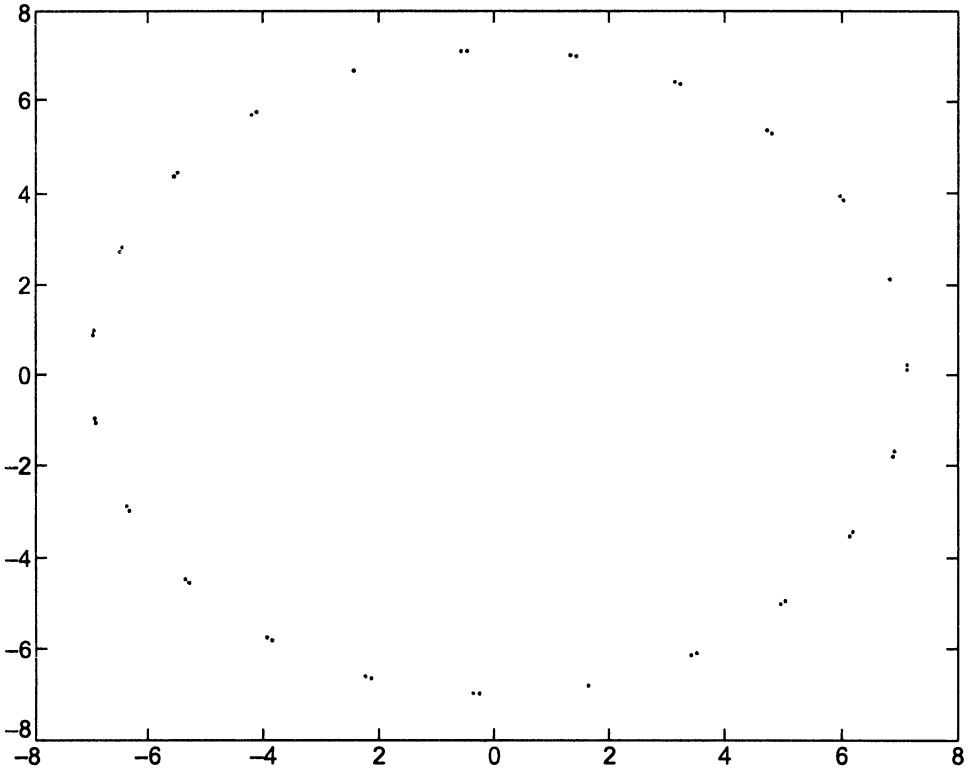

Рис. 2.3. Отображение Пуанкаре для пары гармонических осцилляторов.

графика. Все численные процедуры MATLAB, предназначенные для решения ЗНУ, позволяют применить аппарат локализации событий и соответствующие программные реализации этой возможности полностью аналогичны. В нашем примере мы использовали `ode45` с более жесткими требованиями на допустимые значения ошибок вычислений (по сравнению с соответствующими значениями, принятыми по умолчанию), т. к. необходимо было обеспечить высокую точность вычисления периодического решения при большом значении числа периодов. На рисунке 2.3 показано одно из упомянутых выше отображений Пуанкре, вычисленное с использованием нашей программы. Программа также позволяет вывести дополнительную информацию

```
Event 1 occurred 43 times.
Event 2 occurred 65 times.
```

```
function ch2ex2
opts = odeset('Events',@events,'RelTol',1e-6,'AbsTol',1e-10);
[t,x,te,xe,ie] = ode45(@odes,[0 65],[5; 5; 5; 5],opts);
plot3(x(:,1),x(:,2),x(:,3));
xlabel('x_1(t)'), ylabel('x_2(t)'), zlabel('x_3(t)')
if isempty(ie)
```

```
    fprintf('There were no events.\n');
else
    event1 = find(ie == 1);
    if isempty(event1)
        fprintf('Event 1 did not occur.\n');
    else
        fprintf('Event 1 occurred %i times.\n',length(event1));
        figure
        plot(xe(event1,1),xe(event1,3),'*');
    end
    event2 = find(ie == 2);
    if isempty(event2)
        fprintf('Event 2 did not occur.\n');
    else
        fprintf('Event 2 occurred %i times.\n',length(event2));
        figure
        plot(xe(event2,1),xe(event2,2),'*');
    end
end

%=============================================================
function dxdt = odes(t,x)
a = 3.12121212;
b = 2.11111111;
dxdt = [a*x(3); b*x(4); -a*x(1); -b*x(2)];

function [value,isterminal,direction] = events(t,x)
value = [x(2); x(3)];
isterminal = [0; 0];
direction = [0; 0];
```

ПРИМЕР 2.3.5

В контексте реализации различных моделей локализации событий интересно рассмотреть движение мяча по наклонной плоскости с подпрыгиванием (см. рисунок 2.4). Для простоты изложения предположим, что в качестве наклонной плоскости используется гипотенуза треугольника с координатами вершин $(0,0)$, $(0,1)$ и $(1,0)$. В момент времени t координаты мяча задаются парой $(x(t), y(t))$. Предположим также, что в начальный момент времени мяч расположен выше наклонной плоскости: $x(0) = 0$, $y(0) > 1$, $x'(0) = 0$ и $y'(0) = 0$. При свободном движении мяча его движение описывается уравнениями

$$x'' = 0,\ y'' = -g.$$

Второе уравнение описывает ускорение мяча в вертикальном направлении под воздействием сил гравитации; $g = 9.81$ — гравитационная постоянная. Для того, чтобы записать эти уравнения в виде системы ОДУ первого порядка, введем новые переменные

$$y_1(t) = x(t),\ y_2(t) = x'(t),\ y_3(t) = y(t),\ y_4(t) = y'(t).$$

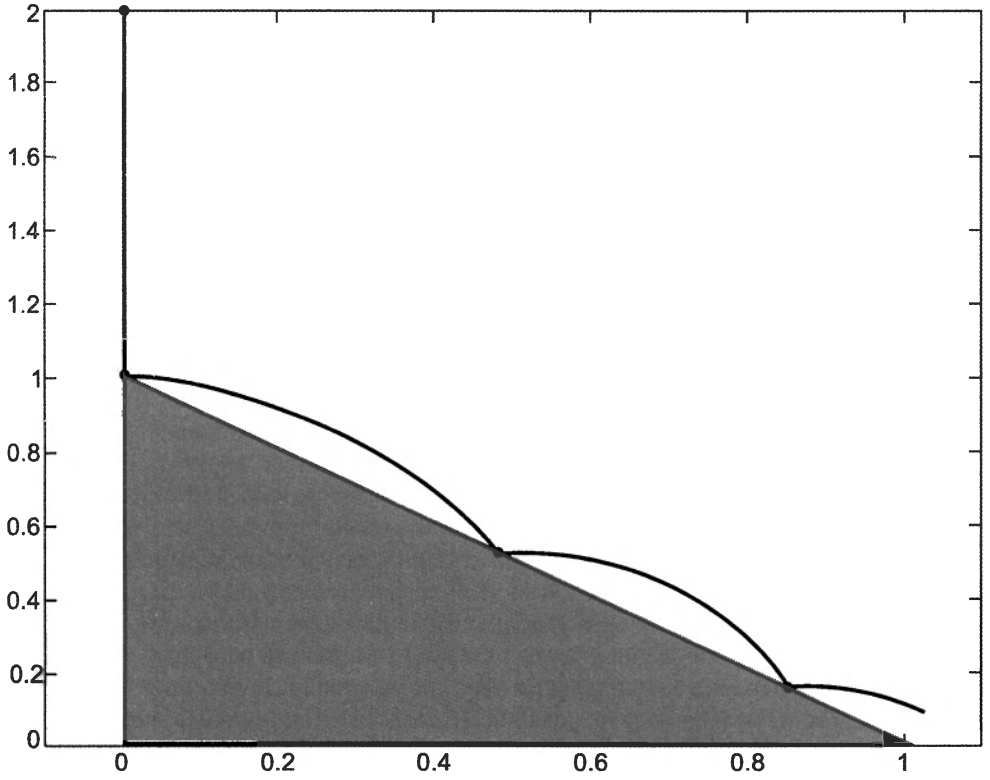

Рис. 2.4. Траектория движения мяча при $k = 0.35$.

Эти уравнения описывают движение мяча до того момента, когда он касается наклонной плоскости в момент времени $t^* > 0$. К моменту касания мяч смещается относительно начального положения вправо на расстояние $x(t^*)$ и в точке касания наклонная плоскость имеет высоту $1 - x(t^*)$. (Первый момент времени, когда возникает событие, соответствующее касанию мяча наклонной плоскости, — это начальный момент времени $x(t^*) = 0$.) В момент касания высота точки расположения мяча равна $y(t)$ и поэтому при касании мяча наклонной плоскости выполнено равенство

$$y(t^*) = 1 - x(t^*).$$

Таким образом, функция обработки события определяется равенством

$$g_1 = y_3 - (1 - y_1).$$

Это событие является терминальным, поскольку в момент касания модель рассматриваемой системы должна измениться. Очень важно выполнить локализацию этого события, т. к. именно это существенным образом определяет дальнейшую траекторию движения мяча. Действительно, в уравнениях движения нет ничего, что повлияло бы на изменение траектории мяча, если бы момент касания был бы пропущен численной процедурой интегрирования этих уравнений. После касания

наклонной плоскости мяч подпрыгивает и его последующее движение описывается другой ЗНУ с теми же уравнениями движения. В качестве начального положения мяча на новом этапе интегрирования используются координаты мяча в точке прерывания вычислительного процесса на предыдущем этапе. Эффект подпрыгивания реализуется тем, что на новом этапе интегрирования начальная скорость ненулевая и направляющий вектор определяется компонентами скорости мяча в момент t^*, а также геометрическими характеристиками наклонной плоскости. Кроме того, диссипация энергии моделируется тем, что абсолютное значение скорости мяча при каждом очередном ударе о наклонную плоскость уменьшается с коэффициентом $0 < k < 1$. Таким образом, начальными данными на новом этапе интегрирования, начинающемся в момент времени t^*, являются

$$(y_1(t^*), -ky_4(t^*), y_3(t^*), ky_2(t^*)).$$

Для локализации момента времени, когда мяч касается наклонной плоскости в очередной раз, используется та же терминальная функция обработки событий. Заметим, что в соответствующей начальной точке очередной ЗНУ терминальная функция обработки события равна нулю. Это объясняет тот факт, что терминальное событие в начальной точке должно обрабатываться специальным образом. Существует другой способ преодоления этой трудности, который в рассматриваемом примере представляется более удобным. Если присвоить выходной переменной `direction` значение 0, численная процедура будет выполнять обработку возникновения событий не только в стандартных терминальных ситуациях, но в начальных точках каждого из этапов интегрирования. Этого можно избежать, присвоив переменной `direction` значение -1. Расстояние мяча от наклонной плоскости увеличивается в момент отскока мяча в начальной точке нового этапа интегрирования и поэтому при указанном значении переменной `direction` численная процедура будет игнорировать возникновение события в этой точке, но обрабатывать это событие в терминальный момент, когда мяч падает на наклонную плоскость.

Вычислительный процесс должен прекратиться в момент, когда мяч достигнет нижнего конца наклонной плоскости, т. е. при $x(t^*) = 1$. Для локализации этого события используется терминальная функция обработки события $g_2 = y_1 - 1$. При некоторых значениях коэффициента k и некоторой начальной высоте расположения мяча над наклонной плоскостью подпрыгивание мяча затухает, вследствие чего он не может достигнуть конца наклонной плоскости. Наша модель движения не описывает подобную ситуацию и мы должны предложить новую модель движения, согласно которой мяч просто скатывается с наклонной плоскости после затухания подпрыгивания. Следовательно, необходимо распознать возникновение события, когда подпрыгивание прекратилось и прервать вычислительный процесс. В нашей программе соответствующее терминальное событие считается наступившим, если два интервала времени между тремя последовательными касаниями мяча наклонной плоскости отличаются друг от друга меньше чем на 1%. Представляет определенные трудности и обработка ситуации, когда мяч касается наклонной плоскости очень близко к ее концу. В этом случае следует прервать вычисления, если длина последнего прыжка меньше 1% расстояния до конца наклонной плоскости.

Некоторые размышления о существе рассматриваемой задачи, а также численные эксперименты, наводят на мысль о том, что при решении этой задачи

могут возникнуть трудности более общего характера. В частности, нам неизвестно, сколько времени потребуется мячу для того, чтобы достигнуть конца наклонной плоскости. Несомненно, длительность интервала интегрирования зависит от начальной высоты мяча и значения коэффициента k. При вызове численных процедур необходимо указывать интервал интегрирования и поэтому нам необходимо оценить его длину. Если наша оценка окажется меньше чем необходимо, мы будем вынуждены продолжить вычисления вручную. Некоторые численные процедуры решения ЗНУ допускают возможность увеличения интервала интегрирования, но в MATLAB это не предусмотрено и используется другой подход, заключающийся в том, что новая ЗНУ может быть решена с начальными данными, являющимися результатами вычислений при предыдущем интегрировании. Этот способ вполне приемлем, поскольку запуск численных процедур MATLAB реализован очень эффективно и достаточно нескольких пробных запусков, чтобы определить необходимую длину интервала интегрирования. В упражнениях 2.27 и 2.28 читателю предлагается решить различные варианты задачи о прыгающем мяче.

Мы бы хотели прокомментировать пару аспектов чисто программистского характера. Траектория мяча состоит из нескольких сегментов, что обусловлено либо соударениями с наклонной плоскостью, либо принятием нами решений об увеличении интервала интегрирования. В этой связи представляется удобным сохранять информацию о всей траектории в двух массивах так, как это сделано в нижеследующей программе. Поскольку рассматриваемая система состоит из простых ОДУ и их численное интегрирование не представляет трудностей, численная процедура выбирает достаточно большие шаги интегрирования, что приводит к тому, что выводимые графики выглядят как ломанные линии. Для решения этой проблемы используется массив `tspan`. Кроме того, при выводе графика решения начальная точка помечается символом «o», а точки контакта с наклонной плоскостью — символом «*».

```
function ch2ex3
% The ball is at (x(t),y(t)).
% Here y(1) = x(t), y(2) = x'(t), y(3) = y(t), y(4) = y'(t).

% Set initial height and coefficient of restitution:
y0 = [0; 0; 2; 0];
k = 0.35;

% Plot the initial configuration:
fill([0 0 1],[0 1 0],[0.8 0.8 0.8]);
axis([-0.1 1.1 0 y0(3)])
hold on
plot(0,y0(3),'ro'); % Mark the initial point.

options = odeset('Events',@events);
% Accumulate the path of the ball in xplot,yplot.
xplot = [];
yplot = [];
tstar = 0;
while 1
```

```
    tspan = linspace(tstar,tstar+1,20);
    [t,y,te,ye,ie] = ode23(@odes,tspan,y0,options);
    % Accumulate the path.
    xplot = [xplot; y(:,1)];
    yplot = [yplot; y(:,3)];
    if isempty(ie)        % Extend the interval.
        tstar = t(end);
        y0 = y(end,:);
    elseif ie(end) == 1   % Ball bounced.
        plot(ye(end,1),ye(end,3),'r*'); % Mark the bounce point.
        if (te(end) - tstar) < 0.01*tstar
            fprintf('Bounces accumulated at x = %g.\n',ye(end,1))
            break;
        end
        if abs(ye(end,1) - 1) < 0.01
            break;
        end
        tstar = te(end);
        y0 = [ye(end,1); -k*ye(end,4); ye(end,3); k*ye(end,2)];
    elseif ie(end) == 2 % Reached end of ramp.
        break;
    end
end
plot(xplot,yplot);

%=================================================
function dydt = odes(t,y)
dydt = [y(2); 0; y(4); - 9.81];

function [value,isterminal,direction] = events(t,y)
value = [ y(3) - (1 - y(1))
            y(1) - 1      ];
isterminal = [1; 1];
direction  = [-1; 0];
```

■ УПРАЖНЕНИЕ 2.23

В работе [Dormand, 1996, стр. 114] рассматривается интересный пример, в котором исследуется процесс извлечения пробки из бутылки с шампанским. Пусть $x(t)$ — смещение пробки в момент времени t и L — длина пробки. При $x(t) \leqslant L$ (пробка находится в горлышке бутылки) величина смещения удовлетворяет равенству

$$\frac{d^2x}{dt^2} = g(1+q)\left[\left(1+\frac{X}{d}\right)^{-\gamma} + \frac{Rt}{100} - 1 + \frac{qx}{L(1+q)}\right],$$

где $g = 9.81, q = 20, d = 5, \gamma = 1.4$ и $R = 4$ — некоторые физические параметры. При начальных данных $x(0) = 0$ и $x'(0) = 0$ и при $L = 3.75$ численно определите момент времени, соответствующий выходу пробки из горлышка бутылки, т. е.

$x(t^*) = L$. Кроме того, вычислите скорость пробки на выходе из бутылки $x'(t^*)$. Эта задача представляет собой пример, в котором следует использовать аппарат локализации событий. Выполните интегрирование ЗНУ до момента наступления терминального события $x(t^*) - L = 0$ и выведите значения t^* и $x'(t^*)$. Единственная сложность, с которой приходится столкнуться при решении этой задачи с использованием численных процедур MATLAB — это необходимость задания интервала интегрирования. Простейшим способом решения этой проблемы является тестовый запуск процедуры интегрирования с интервалом $[0, 100]$. При этом необходимо убедиться, что прекращение вычислений было произведено действительно по причине наступления терминального события. Это может быть сделано либо путем проверки последнего значения независимой переменной, либо путем проверки на наличие хотя бы одного компонента в массиве `ie`. Если не произошло ни одного события, необходимо выполнить вычисления повторно, но с увеличенным интервалом интегрирования.

◘ УПРАЖНЕНИЕ 2.24

В главе 1 рассматривалась система ОДУ, описывающая плоское движение снаряда, выпущенного из пушки

$$\begin{aligned} y' &= \operatorname{tg}(\phi), \\ v' &= -\frac{g\sin(\phi) + \nu v^2}{v\cos(\phi)}, \\ \phi' &= -\frac{g}{v^2}, \end{aligned}$$

где y — высота снаряда над уровнем дульного среза пушки, v — скорость снаряда и ϕ — угол (в радианах) наклона траектории снаряда относительно горизонтальной оси. Независимая переменная x представляет собой расстояние снаряда от пушки. Константа ν служит для моделирования сопротивления воздуха (трения) и $g = 0.032$ — приведенная гравитационная постоянная. При рассмотрении подобной системы естественно исследовать две задачи. Первая заключается в том, что необходимо определить дальность выстрела при заданной начальной скорости снаряда $v(0)$ и начальном угле наклона ствола пушки $\phi(0)$. При начальной высоте снаряда $y(0) = 0$ необходимо локализовать событие, при котором в первый раз $x^* > 0$ величина $y(x)$ обращается в ноль. Легко видеть, что терминальное событие наступает в начальной точке. В главе 1 обсуждалась также и другая задача, сводящаяся к решению ЗГУ, и которую можно сформулировать следующим образом: при каком начальном угле $\phi(0)$ пушка имеет заданную дальность выстрела? На рисунке 1.3 показаны две траектории, полученные для дальности выстрела 5 при значении параметра $\nu = 0.02$ и при начальной скорости $v(0) = 0.5$. В ходе численных экспериментов были определены два значения начального угла $\phi(0) \approx 0.3782$ и $\phi(0) \approx 9.7456$, обеспечивающие решение поставленной задачи. При этих значениях начального угла решите две ЗНУ с заданными значениями $v(0)$ и ν. Для проверки условия близости величины дальности выстрела к заданному значению 5 используйте аппарат локализации событий.

■ УПРАЖНЕНИЕ 2.25

Выполните интегрирование уравнения маятника

$$\theta'' + \sin(\theta) = 0$$

с начальными условиями $\theta(0) = 0$ и $\theta'(0) = 1$ на интервале $[0, 20\pi]$. На указанном интервале периодическое решение совершает несколько циклов колебаний и поэтому для обеспечения большой точности вычислений в качестве значений `RelTol` и `AbsTol` используйте 10^{-5} и 10^{-10}, соответственно. Локализуйте все точки t^*, в которых $\theta(t^*) = 0$. Выведите график $\theta(t)$. Вычислите и выведите график значений разности между последовательными парами значений t^*. В системе MATLAB это может быть сделано с использованием команды `diff(te)`. Поскольку решение является периодическим, при достаточно точном вычислении аппроксимаций решения упомянутые выше значения разностей должны быть приблизительно постоянны. При малых $\theta(t)$ рассматриваемое ОДУ может быть аппроксимировано линейным ОДУ $x'' + x = 0$. Упомянутые выше значения разностей значений t, в которых $x(t)$ обращается в ноль, равны π. Как этот результат соотносится с результатом, полученным для нулей $\theta(t)$? Уменьшите амплитуду колебания $\theta(t)$, уменьшив начальное значение угла наклона $\theta'(0) = 0.1$. Стали ли значения разностей ближе к π? Повторите эксперимент при $\theta'(0) = 0.01$ и прокомментируйте полученный результат.

■ УПРАЖНЕНИЕ 2.26

При исследовании ЗНУ мы обычно получаем таблицу значений решения для заданных значений независимой переменной. Однако в некоторых случаях бывает необходимо получить подобную таблицу значений лишь для одной зависимой переменной. Эта цель может быть достигнута с использованием аппарата локализации событий. В нижеследующем примере, предложенном в работе [Moler & Solomon, 1970], мы покажем, как можно вычислить значения интеграла

$$t(x) = \int_{x_0}^{x} \frac{1}{\sqrt{f(s)}} ds,$$

где $f(x) > 0$ — гладкая функция. Вычисление интеграла эквивалентно следующей ЗНУ

$$\frac{dt}{dx} = \frac{1}{\sqrt{f(x)}}, \quad t(x_0) = 0.$$

При численном решении этого ОДУ могут возникнуть определенные трудности вблизи точки x, в которой $f(x)$ обращается в ноль. В указанной работе замечено, что $x(t)$ удовлетворяет ЗНУ

$$\frac{dx}{dt} = \sqrt{f(x)}, \quad x(0) = x_0,$$

однако вследствие того, что в этом уравнении также присутствует квадратный корень, аналогичные трудности возникают при получении численного решения вблизи точки $x(t)$, в которой $f(x(t))$ меняет свой знак, или (эквивалентно), где $x'(t)$

обращается в ноль. Для того, чтобы разрешить эту проблему, авторы предложили продифференцировать это уравнение и рассмотреть следующую ЗНУ

$$x'' = 0.5f'(x),\ x(0) = x_0,\ x'(0) = \sqrt{f(x_0)}.$$

Численное интегрирование этого уравнения не вызывает сложности при прохождении траектории решения через точку, в которой $x'(t)$ обращается в ноль и поэтому эта точка может быть легко локализована в процессе вычислений путем использования соответствующего терминального события. Задача состоит в том, чтобы получить значения t_i, в которых $x(t)$ равно заданным значениям x_i, т. е. необходимо определить моменты времени, в которых функции $x(t) - x_i$ обращаются в ноль и это требование определяет набор необходимых нетерминальных событий. Напишите программу, позволяющую получить таблицу значений интеграла для заданных x_i. Вы можете задать значения x_i непосредственно в тексте функции обработки событий, однако более гибкий подход заключается в том, чтобы глобально определить в главной программе массив `xvalues` значений x_i и использовать его в функции обработки событий. Проверьте правильность написания программы, получив значения интеграла для $x_0 = 0$ и `xvalues = 0.1:0.1:0.9` при $f(x) = x$. Используйте при этом численную процедуру `ode45`, выполнив интегрирование на интервале $0 \leqslant t \leqslant 2$. Покажите аналитически, что решение имеет вид $t(x) = 2\sqrt{x}$ и сравните значения, полученные с использованием этого результата, с полученными вами численными значениями. Тот факт, что $f(x)$ обращается в ноль в начальной точке, не вызывает трудностей при численном интегрировании, но следует иметь в виду, что в этой начальной точке наступает терминальное событие. Напомним, что подобная ситуация является специальным случаем и при локализации события численная процедура сообщает о наступлении этого события, но не прерывает вычислительный процесс. После проверки работоспособности программы указанным способом модифицируйте ее так, чтобы получить таблицу значений интеграла для $f(x) = 1 - x$ и $f(x) = 1 - x^2$.

■ УПРАЖНЕНИЕ 2.27

С использованием программы `ch2ex3.m` можно при различных значениях коэффициента $k = 0.35$ получить траекторию движения мяча, который подпрыгивая скатывается с наклонной плоскости. Выполните численные эксперименты при различных значениях k. В частности, уменьшите k до такой величины, при которой наблюдается подпрыгивание мяча на месте. Например, выполните эксперименты при $k = 0.6$ и $k = 0.3$.

■ УПРАЖНЕНИЕ 2.28

Предположим, что в ситуации, описанной в примере 2.3.5, в конце наклонной плоскости, т. е. в $x(t) = 1$, установлена вертикальная стенка. Модифицируйте программу `ch2ex3.m` для получения траекторий мяча в подобной ситуации. Наиболее простым способом отображения стенки на рисунке является использование команды `axis([-0.1 1 0 y0(3)])`, в результате чего правый край графика станет играть роль этой стенки. При отсутствии стенки вычисленная траектория обрывается при достижении конца наклонной плоскости, что может быть установлено путем проверки условия `ie(end) == 2`. В рассматриваемой в этом упражнении

ситуации это событие соответствует соударению со стенкой. После возникновения этого события продолжите интегрирование с новыми начальными данными

$$(y_1(t^*), -ky_2(t^*), y_3(t^*), y_4(t^*)).$$

Если мяч попадает приблизительно в угол, т. е. при $y_3(t^*) \leqslant 0.01$, прервите вычислительный процесс. Постройте траектории мяча при $k = 0.35$ и $k = 0.7$.

■ УПРАЖНЕНИЕ 2.29

Аппарат локализации событий используется в программе `dfs.m` для прерывания процесса интегрирования, если $y(t)$ достигает верхней или нижней границы графика. Следует отметить, что представленный способ достижения этой цели не единственный. Процедура построения графика отображает лишь часть траектории $y(t)$, лежащую внутри заданного окна. Поэтому для прекращения вычислений можно потребовать, чтобы численная процедура прерывалась при выходе решения за границы этого окна. Это может быть реализовано в функции вывода. Сделайте копию `dfs.m` и дайте этому файлу другое имя. Удалите из текста этой программы функцию обработки событий и модифицируйте функцию вывода так, чтобы вычислительный процесс прерывался, если `y(end) < wB`, или если `y(end) > wT`. Этот подход более прост и эффективен по сравнению с использованием аппарата локализации событий, поскольку при этом не приходится выполнять трудоемкие операции, связанные с точным определением моментов времени t^*, при которых траектория $y(t)$ достигает верхней или нижней границы окна.

2.3.2. ОДУ в форме представления с матрицей весовых коэффициентов

Некоторые численные процедуры допускают представление решаемой системы ОДУ не только в стандартной $y' = f(t, y)$, но и в более общей форме. В частности, численные процедуры MATLAB, предназначенные для решения ЗНУ, допускают представление системы в виде

$$M(t, y)y' = f(t, y), \tag{2.37}$$

где $M(t, y)$ — матрица весовых коэффициентов. Если матрица $M(t, y)$ сингулярна, то такая система является системой *алгебро-дифференциальных уравнений* (АДУ). В этой книге мы не рассматриваем методы решения АДУ и отметим лишь то, что численные процедуры MATLAB, предназначенные для решения жестких ОДУ, позволяют решить АДУ указанного вида [Shampine, Reichelt & Kierzenka, 1999]. В подпараграфе 2.3.3 мы рассмотрим метод численного решения дифференциальных уравнений в частных производных ДУЧП, который позволяет представить их в виде системы ОДУ. В некоторых случаях этот подход приводит к системе вида (2.37). В примере 2.3.11 эти вопросы рассматриваются более детально.

В простейшем случае для решения в MATLAB систем ОДУ в форме представления (2.37) с матрицей весовых коэффициентов $M(t, y)$ при вызове соответствующих численных процедур необходимо использовать опцию `Mass`. Все остальные входные и выходные параметры сохраняют тот смысл, который они имеют при численном решении системы ОДУ, представленной в стандартной форме. Если матрица весовых коэффициентов является постоянной, то при вызове численной

процедуры она должна использоваться в качестве значения опции `Mass`. Это удобно для пользователя и эффективно с вычислительной точки зрения. Предположим, например, что для решения вашей задачи вы выбрали явный метод Рунге–Кутта и рассматриваемая система имеет вид (2.37). В этом случае метод Рунге–Кутта будет в действительности применен к системе ОДУ, представленной в стандартной форме

$$y' = F(t, y) \equiv M^{-1} f(t, y).$$

При инициализации численной процедуры вычисляется LU-разложение для матрицы M. Кроме того, во всех случаях, когда необходимо вычислить $F(t, y)$, будет выполняться вычисление $f(t, y)$ и решение линейной системы для F, с использованием вычисленного и сохраненного в памяти компьютера LU-разложения для матрицы M. В подобных ситуациях возможность использования матрицы весовых коэффициентов является лишь удобной для пользователя формой представления данных. Однако существуют численные процедуры, в которых присутствие матрицы весовых коэффициентов приводит к модификации используемого в них алгоритма интегрирования. Не вдаваясь в детали, можно сказать, что при реализации численных процедур, основанных например на МДН, в качестве итерационной матрицы используется не $I - h\gamma J$, как в стандартном случае, а $M - h\gamma J$. Поскольку для итерационной матрицы в любом случае требуется вычислять LU-разложение, дополнительные вычислительные затраты, связанные с более общей формой представления решаемой системы ОДУ (2.37), нивелируются. Если матрица весовых коэффициентов не является постоянной, используемый алгоритм численного интегрирования подвергается еще более существенной модификации. В этом случае необходимо написать специальную функцию, в которой вычисляется матрица весовых коэффициентов, и при вызове соответствующей численной процедуры в качестве значения опции `Mass` передать указатель на эту функцию.

При решении ОДУ в форме (2.37) вы должны задать матрицу весовых коэффициентов и проинформировать численную процедуру о том, является ли эта матрица сингулярной в начальной точке (т. е. является ли решаемая система ОДУ в действительности системой АДУ). Для этого используется опция `MassSingular`, которая может принимать три значения: `yes`, `no` и принятое по умолчанию значение `maybe`. В последнем случае численная процедура выполняет численный тест на сингулярность, что можно избежать, если указать для этой опции одно из первых двух значений. При численном решении больших систем очень важно учесть разреженность матрицы весовых коэффициентов $M(t, y)$ и проинформировать численную процедуру о характере зависимости этой матрицы от y. Более подробно этот вопрос исследуется в примере 2.3.11.

ПРИМЕР 2.3.6

В MATLAB 6 представлена основанная на примере 4.3A из работы [Wells, 1967] демонстрационная программа `batonode`, иллюстрирующая интегрирование системы ОДУ, представленной в форме (2.37). Мы рассмотрим этот пример более подробно, поскольку в документации MATLAB он не рассматривается. В примере исследуется свободное движение палочки в вертикальной плоскости под воздействием гравитационного поля. Палочка моделируется двумя массами m_1 и m_2,

соединенными невесомым стержнем длины L. Если в момент времени t координаты первой массы $(X(t), Y(t))$ и угол наклона палочки по отношению к горизонтальной оси равен $\theta(t)$, то уравнения движения в форме Лагранжа имеют вид (2.37) с матрицей весовых коэффициентов, зависящей от $\theta(t)$. Эти ОДУ могут быть представлены в векторной форме, если ввести вектор

$$y = (X, X', Y, Y', \theta, \theta')^T.$$

Соответствующие функции $f(t, y)$ и $M(t, y)$ определены в нижеприведенном тексте программы `batonode`. (Для краткости изложения в представленном тексте программы не приведены комментарии, а также команды, обеспечивающие графический вывод результатов моделирования.) Эта простая, не жесткая задача может быть численно решена непосредственно в форме представления с матрицей весовых коэффициентов. В этой программе используется возможность передачи значений параметров функций, определяющих решаемую систему ОДУ и матрицу весовых коэффициентов, в виде параметров численной процедуры интегрирования — указанные параметры перечислены в конце ее списка аргументов. Эта возможность реализована во всех функциях MATLAB и рассматриваемый пример может служить иллюстрацией ее применения. В качестве значения опции `MassSingular` может быть использовано `no`, но с учетом того, что система состоит всего из шести уравнений, можно его не задавать, т. к. в рассматриваемом случае проверка сингулярности вычислительно малозатратна. По той же причине можно не принимать во внимание фактор разреженности матрицы, но для его учета достаточно изменить команду `M = zeros(6,6)` на `M = sparse(6,6)`. На рисунке 2.5 показаны результаты моделирования движения палочки, полученные с использованием программы `batonode`. (В упражнении 2.31 читателю предлагается выполнить с использованием программы `batonode` дополнительные эксперименты.)

```
function batonode
.
.
.
m1 = 0.1;
m2 = 0.1;
L = 1;
g = 9.81;

tspan = linspace(0,4,25);
y0 = [0; 4; 2; 20; -pi/2; 2];

options = odeset('Mass',@mass);
[t y] = ode45(@f,tspan,y0,options,m1,m2,L,g);
.
.
.
% ------------------------------------------

function dydt = f(t,y,m1,m2,L,g)
dydt = [
```

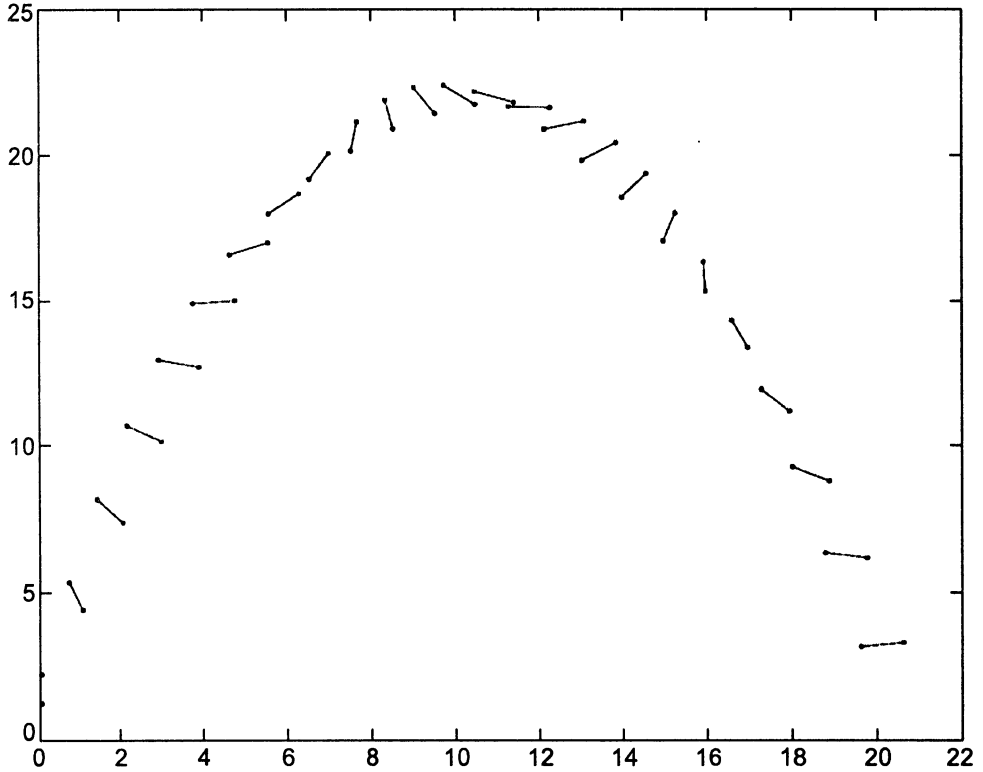

Рис 2.5. Движение подброшенной палочки, описываемое системой ОДУ с матрицей весовых коэффициентов.

```
 y(2)
 m2*L*y(6)^2*cos(y(5))
 y(4)
 m2*L*y(6)^2*sin(y(5))-(m1+m2)*g
 y(6)
 -g*L*cos(y(5))
 ];
% ------------------------------------------------

function M = mass(t,y,m1,m2,L,g)
M = zeros(6,6);
M(1,1) = 1;
M(2,2) = m1 + m2;
M(2,6) = -m2*L*sin(y(5));
M(3,3) = 1;
M(4,4) = m1 + m2;
M(4,6) = m2*L*cos(y(5));
M(5,5) = 1;
M(6,2) = -L*sin(y(5));
```

```
M(6,4) = L*cos(y(5));
M(6,6) = L^2;
```

ПРИМЕР 2.3.7

Нестационарное уравнение Навье–Стокса для одномерного потока несжимаемой жидкости, заданное на интервале длины L, может быть представлено в виде дифференциального уравнения в частных производных (ДУЧП)

$$\frac{\partial U}{\partial t} + A\frac{\partial U}{\partial z} = C,$$

заданного при $t \geqslant 0$ на интервале $0 \leqslant z \leqslant L$. Компонентами вектора $U = (\rho\ G\ T)^T$ являются плотность ρ, скорость потока G и температура T. В вышеприведенной системе ДУЧП используются вектор

$$C = \begin{pmatrix} 0 \\ -KG\left|\dfrac{G}{\rho}\right| - \rho g_a \sin(\theta) \\ \dfrac{a^2 \Phi P_H \kappa}{C_p A_f} \end{pmatrix}$$

и матрица

$$A = \begin{pmatrix} 0 & 1 & 0 \\ \dfrac{1}{\rho\kappa} - \dfrac{G^2}{\rho^2} & 2\dfrac{G}{\rho} & \dfrac{\beta}{\kappa} \\ -\dfrac{a^2\beta\bar{T}G}{\rho^2 C_p} & \dfrac{a^2\beta\bar{T}}{\rho C_p} & \dfrac{G}{\rho} \end{pmatrix},$$

где $\bar{T} = T + 273.15$. Свойства потока и физические параметры описываются в работе [Thompson & Tuttle, 1986]. В тексте программы `ch2ex4.m` задаются постоянные значения этих величин, а также определяются другие переменные, идентификаторы которых соответствуют почти очевидным образом тем обозначениям, которые используются в вышеприведенных уравнениях (например, `sinth` соответствует $\sin(\theta)$). Рассмотрим граничные условия

$$\rho(0,t) = \rho_0 = 795.5,$$
$$T(0,t) = T_0 = 255.0,$$
$$G(L,t) = G_0 = 270.9.$$

Задача состоит в том, чтобы вычислить стационарное решение этих уравнений. В установившемся режиме выполнено $\rho_t = 0$. При этом $G(z)$ является постоянной матрицей G_0 на всем интервале $0 < z < L$ и рассматриваемая система ДУЧП сводится к системе ОДУ

$$\begin{pmatrix} \dfrac{1}{\rho\kappa} - \dfrac{G_0^2}{\rho^2} & \dfrac{\beta}{\kappa} \\ -\dfrac{a^2\beta\bar{T}G_0}{\rho^2 C_p} & \dfrac{G_0}{\rho} \end{pmatrix} \begin{pmatrix} \dfrac{d\rho}{dz} \\ \dfrac{dT}{dz} \end{pmatrix} = \begin{pmatrix} -KG_0\left|\dfrac{G_0}{\rho}\right| - \rho g_a \sin(\theta) \\ \dfrac{a^2 \Phi P_H \kappa}{C_p A_f} \end{pmatrix}.$$

Преобразование системы ДУЧП к системе ОДУ, выполненное в соответствии с вышеприведенной процедурой, позволяет получить решение, которое иногда называют *решением для непрерывного пространства и дискретного времени (НПДВ-решение)* (continuous space, discrete time solution). В зависимости от граничных условий для исходной системы ДУЧП, этот подход приводит к рассмотрению либо некоторой ЗНУ для системы ОДУ, либо ЗГУ. Полученные уравнения часто используются при моделировании переохлажденной жидкости в трехфазном парогенераторе энергетической системы. (Аналогичные, но более сложные уравнения могут быть получены для зон насыщения пар-жидкость и чистого пара.) В подобных моделях параметры (движения) границ между зонами определяются свойствами уравнений состояния. Например, система ОДУ может быть проинтегрирована из начального состояния $z = 0$ в положительном по z направлении до тех пор, пока плотность ρ не станет равной плотности насыщения «со стороны жидкости» $\rho_{sat}(T)$. Как описано в параграфе 2.3.1, мы можем локализовать это событие с использованием функции обработки события

$$g(z, \rho, T) = \rho(z) - \rho_{\mathtt{sat}}(T(z)).$$

В чисто иллюстративных целях в программе моделирования используется следующее уравнение состояния

$$\rho_{\mathtt{sat}}(T) = -3.3(T - 290) + 738.$$

Даже с учетом существенного упрощения модели, соответствующая ЗНУ является непростой. Однако ее численное решение в MATLAB не представляет труда, поскольку соответствующие численные процедуры допускают решение систем, представленных в терминах матрицы весовых коэффициентов, а также позволяют использовать при необходимости аппарат локализации событий. При вызове `ch2ex4.m` на экране отображается следующая строка

```
Upper boundary at z = 2.09614
```

а также график, изображенный на рисунке 2.6. В упражнении 2.32 рассмотренная в этом примере модель стационарного НПДВ-решения используется для исследования потока жидкости на входе насоса при его отключении.

```
function ch2ex4
% Define the physical constants:
global kappa beta a Cp K ga sinth Phi Ph Af G0
kappa = 0.171446272015689e-8;
beta  = 0.213024626664637e-2;
a     = 0.108595374561510e+4;
Cp    = 0.496941623289027e+4;
K     = 10;
ga    = 9.80665;
sinth = 1;
Phi   = 1.1e+5;
Ph    = 797.318;
Af    = 3.82760;
```

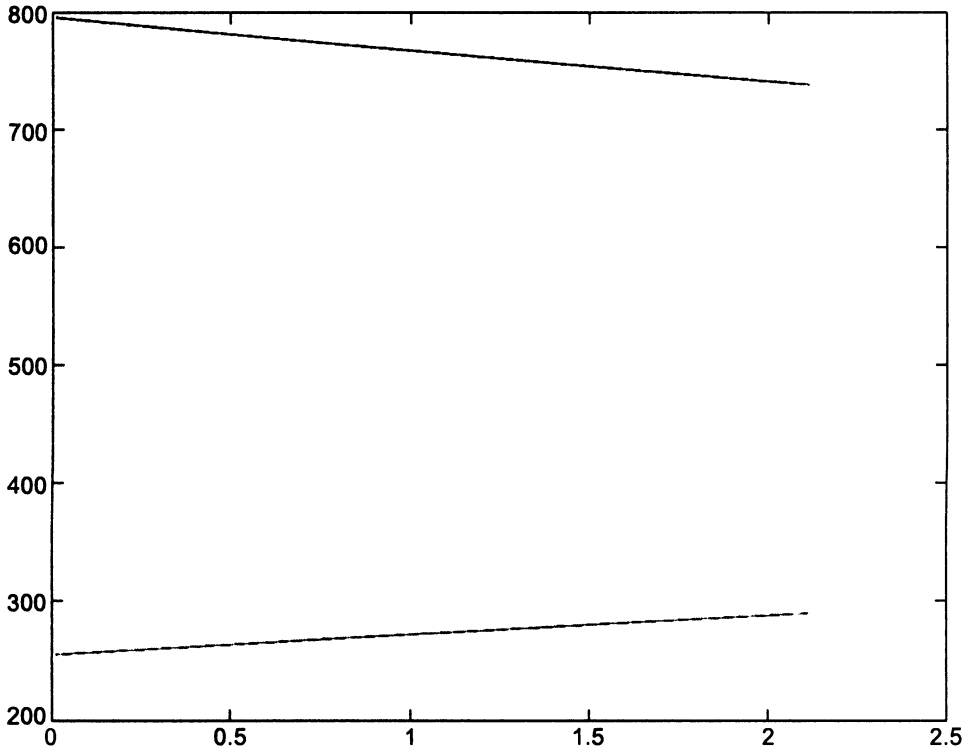

Рис. 2.6. Установившееся решение для верхней границы жидкости.

```
G0    = 270.9;

options = odeset('Mass',@mass,'MassSingular','no','Events',@events);
[z,y,ze,ye,ie] = ode45(@odes,[0 5],[795.5; 255.0],options);
if ~isempty(ie)
fprintf('Upper boundary at z = %g.\n',ze(end));
end
plot(z,y);

%=====================================================================
function dydz = odes(z,y)
global kappa beta a Cp K ga sinth Phi Ph Af G0
rho = y(1);
T = y(2);
dydz = [ (-K*G0*abs(G0/rho) - rho*ga*sinth)
         (a^2 *Phi*Ph*kappa)/(Cp*Af)        ];

function A = mass(z,y)
global kappa beta a Cp K ga sinth Phi Ph Af G0
rho = y(1);
T = y(2);
```

```
A = zeros(2);
A(1,1) = 1/(rho*kappa) - (G0/rho)^2;
A(1,2) = beta/kappa;
A(2,1) = -(a^2 *beta*(T + 273.15)*G0)/(Cp*rho^2);
A(2,2) = G0/rho;

function [value,isterminal,direction] = events(z,y)
isterminal = 1;
direction  = 0;
rho    = y(1);
T      = y(2);
rhosat = -3.3*(T - 290.0) + 738.0;
value  = rho - rhosat;
```

■ УПРАЖНЕНИЕ 2.30

Рассмотренная в упражнении 2.19 модель термического разложения озона представлена в форме, которая удобна при применении методов сингулярных возмущений и содержит матрицу весовых коэффициентов $M = \mathrm{diag}\{1, \varepsilon\}$. Выполните упражнение 2.19 с использованием численных методов интегрирования системы ОДУ, представленной в терминах матрицы весовых коэффициентов. При написании программы примите во внимание, что матрица M является постоянной.

■ УПРАЖНЕНИЕ 2.31

В демонстрационной программе `batonode` в качестве значений длины стержня используется 1, а массы обоих тел — 0.1. Сделайте копию этой программы и модифицируйте ее текст так, чтобы можно было определить движение палочки при новых значениях этих констант: длина $L = 2$ и масса второго тела `m2` равна 0.5. При этом необходимо изменить параметры координатных осей при помощи функции `axis`. Какие наблюдаются отличия в качественных характеристиках движения палочки при новых значениях констант?

■ УПРАЖНЕНИЕ 2.32

Стандартным применением модели стационарного НПДВ-решения, рассмотренной в примере 2.3.7, является исследование процесса экспоненциального убывания потока жидкости во входном клапане насоса при его выключении. Найдите НПДВ-решение при `tD = linspace(0, 1, 11)`, если скорость потока во входном клапане в моменты времени `tD(i)` определяется равенством

$$G_0 = 270.9(0.8 + 0.2e^{-\mathtt{tD(i)}}).$$

Поставленная задача может быть решена путем модификации программы `ch2ex4.m`, выполненной таким образом, чтобы можно было в цикле получить значения уровня верхней границы `zU(i)` жидкости при указанном значении G_0. Поскольку в программе `ch2ex4.m` уже определена глобальная переменная для скорости потока, необходимо ввести новую переменную в теле цикла перед вызовом `ode45`. Для графического отображения графика уровня верхней границы как функции от времени используйте команду `plot(tD,zU)`.

■ УПРАЖНЕНИЕ 2.33

Двойной маятник состоит из двух соединенных между собой простых маятников. Пусть $\theta_1(t)$ и $\theta_2(t)$ — углы отклонения верхнего и нижнего сегментов маятника от вертикальной оси; m_i — массы, расположенные на концах сегментов; L_i — длины сегментов. Если на движение маятника оказывает воздействие только сила гравитации, уравнения движения имеют следующий вид [Borrelli & Coleman, 1999]

$$(m_1+m_2)L_1\theta_1'' + m_2L_2\cos(\theta_2-\theta_1)\theta_2'' - \\ -m_2L_2(\theta_2')^2\sin(\theta_2-\theta_1) + (m_1+m_2)g\sin(\theta_1) = 0,$$
$$m_2L_2\theta_2'' + m_2L_1\cos(\theta_2-\theta_1)\theta_1'' + m_2L_1(\theta_1')^2\sin(\theta_2-\theta_1) + m_{2g}\sin(\theta_2) = 0.$$

В работе [Giordano & Weir, 1991] рассмотрена модель вертолета Chinook с двумя грузовыми поддонами, подвешенными под вертолетом на стропах. При некоторой идеализации в качестве модели движения этих поддонов можно использовать приведенную выше модель двойного маятника. При моделировании использовались следующие численные значения: $m_1 = 937.5$ и $m_2 = 312.5$ слаг[1] — массы верхнего и нижнего поддонов, соответственно; $L_1 = L_2 = 16$ футов — длины строп; $g = 32$ фут/с^2 — ускорение свободного падения. Крюк, поддерживающий нижний поддон, является не защелкивающимся и поэтому если нижний поддон при колебаниях отклоняется более чем на $\pi/3$ радиан, строп может сорваться с крюка и поддон будет потерян. При быстрых маневрах вертолета поддоны приходят в движение, которое может быть промоделировано с использованием представленной модели и при начальных данных

$$\theta_1(0) = -0.5, \quad \theta_1'(0) = -1, \quad \theta_2(0) = 1, \quad \theta_2'(0) = 2.$$

Представленные дифференциальные уравнения могут быть представлены в терминах матрицы весовых коэффициентов и непосредственно решены с использованием `ode45` при значениях допустимых ошибок вычислений, принятых по умолчанию. Выполните интегрирование на промежутке от $t = 0$ до $t = 2\pi$, но прекратите вычислительный процесс, если функция обработки события $\theta_2(t) - \pi/3$ обращается в ноль (т. е. если нижний поддон потерян). В указанной работе использовались линеаризованные уравнения

$$(m_1+m_2)L_1\theta_1'' + m_2L_2\theta_2'' + (m_1+m_2)g\theta_1 = 0,$$
$$m_2L_2\theta_2'' + m_2L_1\theta_1'' + m_2g\theta_2 = 0,$$

поскольку в этом случае нетрудно найти аналитическое решение. Используемые численные значения параметров удовлетворяют соотношениям $g = 2L_1 = 2L_2$ и $m_1 = 3m_2$ и поэтому аналитическое решение имеет простой вид

$$\theta_1(t) = -\tfrac{1}{2}\cos(2t) - \tfrac{1}{2}\sin(2t),$$
$$\theta_2(t) = \cos(2t) + \sin(2t).$$

Найдите численное решение представленной линейной модели и сравните полученный результат с аналитическим решением. Кроме того, сравните результаты

[1] Прим. перев. — Слаг (англ. slug), единица массы в системе англ. мер 1 слаг =14.5939 кг.

моделирования линейной и нелинейной модели, обратив особое внимание на значение момента времени, когда нижний поддон срывается с крюка. Результаты этого сравнения должны быть очень близки.

2.3.3. Большие системы и метод прямых

В этом параграфе будут рассмотрены вопросы, которые имеют важное значение при решении ЗНУ для больших систем уравнений. Среда программирования MATLAB не предназначена для численного анализа *очень* больших систем, выполняемого при некоторых научных исследованиях, но вполне может быть использована при решении достаточно широкого класса систем. *Метод прямых (МП)* позволяет аппроксимировать систему дифференциальных уравнений в частных производных (ДУЧП) системой обыкновенных дифференциальных уравнений (ОДУ). В результате применения этого метода исходная задача решения системы ДУЧП сводится к решению сравнительно большой и возможно жесткой системы ОДУ. МП широко применяется при решении ДУЧП, но в этом параграфе мы рассмотрим этот метод в контексте задачи численного решения больших систем ОДУ. В системе MATLAB имеется численная процедура `pdepe`, основанная на использовании МП и предназначенная для решения небольших систем параболических и эллиптических ДУЧП относительно одной пространственной переменной и независимой переменной времени. Кроме того, в разработанном для MATLAB наборе программ `PDE Toolbox` метод прямых используется для решения системы ДУЧП относительно двух пространственных переменных и переменной времени.

Идея метода прямых заключается в дискретизации всех переменных ДУЧП за исключением одной переменной, в результате чего получается система ОДУ. Этот подход часто называют *полудискретизацией* (semidiscretization). Обычно дискретизируются пространственные переменные, а переменная времени используется в качестве непрерывной переменной. Для дискретизации пространственных переменных применяют различные методы. Применение двух из них рассматривается в нижеследующем примере.

ПРИМЕР 2.3.8

Одностороннее волновое уравнение в частных производных (one-way wave) (известное также как уравнение адвекции или уравнение конвекции)

$$u_t + c(x)u_x = 0, \quad c(x) = \tfrac{1}{5} + \sin^2(x-1)$$

рассматривается на $0 \leqslant x \leqslant 2\pi$ и $0 \leqslant t \leqslant 8$. В работе [Trefethen, 2000] это уравнение исследовалось при начальных значениях $u(x,0) = e^{-100(x-1)^2}$ и периодических граничных условиях $u(0,t) = u(2\pi,t)$. Начальное сечение $u(x,0)$ не является периодической функцией, но она убывает вблизи концов интервала настолько быстро, что может рассматриваться как периодическая. Кроме того, решениями этого ДУЧП являются волны, которые распространяются направо. Из рисунка 2.7 видно, что для указанного интервала времени расположенный первоначально в точке $x = 1$ пик не достигает границы рассматриваемой области. Заметим, что граничное условие должно быть задано для левого конца интервала, а для правого конца интервала соответствующее граничное условие не задается. При использовании МП на интервале $[0, 2\pi]$ выбирается сетка $x_1 < x_2 < \ldots < x_N$. В качестве аппрок-

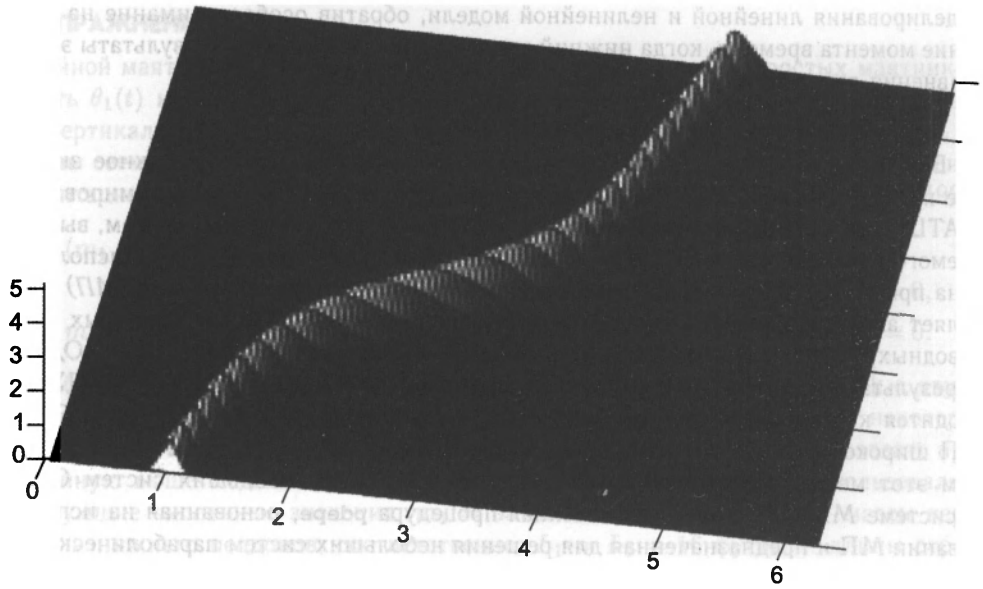

Рис. 2.7. Решение $u_t + c(x)u_X = 0$ с использованием метода прямых.

симаций $u(x_m, t)$, $m = 1, 2, \ldots, N$ используются функции $v_m(t)$, определяемые системой ОДУ

$$\frac{dv_m}{dt} = -c(x_m)(Dv)_m, \quad m = 1, 2, \ldots, N$$

с начальными значениями

$$v_m(0) = u(x_m, 0).$$

Если $(Dv)_m \approx u_x(x_m, t)$, то это уравнение очевидно является аппроксимацией исходного ДУЧП. Небольшая, но очень хорошая книга [Trefethen, 2000] посвящена спектральным методам. Необходимые при их использовании частные производные аппроксимируются в точке (x_m, t) следующим образом: по значениям $v_1(t), v_2(t), \ldots, v_N(t)$ строится интерполяционный тригонометрический полином для x, после чего этот полином дифференцируется по x, и полученная производная вычисляется в x_m. Таким образом, аппроксимация значения этой производной вычисляется с учетом данных на всем рассматриваемом интервале и это обстоятельство является характеристической особенностью спектрального метода. Если точки сетки выбраны равноразнесенными, полученная в соответствии с представленной процедурой аппроксимация позволяет при анализе системы использовать быстрое преобразование Фурье (БПФ). Детальное описание применения БПФ обсуждается в указанной книге при рассмотрении программы `p6`. Наша программа `ch2ex5.m` отличается от программы `p6` тем, что для интегрирования ОДУ вместо многошагового метода с постоянным шагом используется численная процедура `ode23`. Разумеется, с точки зрения вычислительных затрат использование стандартной численной процедуры приводит к некоторым издержкам, но при этом мы избегаем проблем, связанных с выполнением первых шагов при использовании многошагового метода, а также избавляем себя от необходимости подбора длины

шага интегрирования и контроля величины ошибок вычислений в процессе интегрирования. Кроме того, время необходимое для написания новой качественной программы несравнимо велико по сравнению с временем, необходимым для интегрирования уравнения с использованием стандартной и отлаженной программы!

```
function ch2ex5
global ixindices mc

% Spatial grid, initial values:
N  = 128;
x  = (2*pi/N)*(1:N);
v0 = exp(-100*(x - 1) .^ 2);

% Quantities passed as global variables to f(t,v):
ixindices = i * [0:N/2-1 0 -N/2+1:-1]';
mc = - (0.2 + sin(x - 1) .^ 2)';

% Integrate and plot:
t = 0:0.30:8;
[t,v] = ode23(@f,t,v0);
mesh(x,t,v), view(10,70), axis([0 2*pi 0 8 0 5])
ylabel t, zlabel u, grid off

%=============================================================
function dvdt = f(t,v)
global ixindices mc
dvdt = mc .* real(ifft(ixindices .* fft(v)));
```

При использовании метода с постоянным шагом (см., например, [Trefethen, 2000]) основная проблема заключается в выборе достаточно малого шага переменной времени, обеспечивающего достаточно точное вычисление решения. При использовании численных процедур типа `ode23` возникает другая проблема, связанная с выбором точек, в которых требуется вывести значения аппроксимации решения, т. к. численная процедура выбирает шаг интегрирования с учетом заданной точности вычислений. Разумеется, можно довольствоваться стандартной процедурой, которая осуществляет вывод значения решения в конце каждого шага, но в этом случае при интегрировании больших систем может потребоваться значительный объем оперативной памяти. Объем выводимой информации может быть уменьшен, если задать точки, в которых необходимо получить аппроксимацию решения. Более того, с использованием специально написанной функции вывода можно получить полный контроль над потоком вывода (см. пример 2.3.3). В программе `ch2ex5.m` для получения вполне приемлемого графика и несравнимо меньшего по объему файла с выходными данными нам оказалось достаточно вывести лишь половину того количества точек, которые были получены с использованием программы `p6`. Отметим, что для обеспечения выполнения аналогичных требований относительно объема выходных данных при вызове стандартной численной процедуры решения ДУЧП `pdepe` требуется не только задать фиксированную

сетку для пространственных переменных, но и указать моменты времени, в которые необходимо осуществлять вывод данных. В подобных ситуациях не следует выполнять вывод выходных данных в виде структуры-решения, поскольку эта структура содержит аппроксимации значений решения, полученные на каждом шаге интегрирования, а также значительный объем дополнительной информации, необходимой для вычисления непрерывного решения.

Многие полагают, что любая система ОДУ среднего или большого порядка, полученная в результате применения метода прямых, является жесткой. Однако в общем случае это не так. Система из 128 ОДУ, полученная при спектральной дискретизации рассмотренного выше волнового уравнения, является умеренно жесткой. В этом можно убедиться, если вместо численной процедуры `ode23`, предназначенной для решения нежестких задач, использовать численную процедуру `ode15s`: использование численной процедуры, специально предназначенной для решения жестких ЗНУ, приводит к *увеличению* времени интегрирования более чем в пять раз. Таким образом, если высокая степень устойчивости лежащего в основе `ode15s` метода не является определяющим фактором при выборе численной процедуры, следует использовать какую-либо другую численную процедуру, предназначенную для решения нежестких задач. По умолчанию матрица Якоби аппроксимируется в `ode15s` численно. С вычислительной точки зрения эта операция весьма затратна, поскольку в случае системы N уравнений необходимо N раз вычислить значение функции $f(t,v)$ (возможно потребуется выполнить дополнительные вычисления этой функции, если возникнут сомнения в точности полученной аппроксимации). Мы уже отмечали в этой связи, что реализация численной процедуры `ode15s` наиболее эффективна среди всех остальных процедур, предназначенных для решения жестких систем, поскольку значения матриц Якоби сохраняются в памяти компьютера и дополнительные вычисления этих матриц выполняются лишь тогда, когда это необходимо для эффективного вычисления неявных формул. При решении нежестких задач шаг интегрирования обычно настолько мал, что достаточно вычислить всего лишь несколько матриц Якоби. Для исследования этого вопроса можно задать значение опции `Stats` равным `on` и выполнить модифицированную программу `ch2ex5.m`. В результате получаем

```
>> ch2ex5
199 successful steps
6 failed attempts
328 function evaluations
1 partial derivatives
34 LU decompositions
324 solutions of linear systems
```

Заметим, что для вычисления неявных формул на одном шаге интегрирования потребовалось вычислить функцию f очень малое количество раз. Численная процедура вычислила матрицу Якоби (частную производную) один раз и использовала соответствующее значение все 34 раза, когда было необходимо вычислить LU-разложение итерационной матрицы. С учетом этой статистики можно утверждать, что численная процедура `ode15s` очень эффективна при решении нежестких ЗНУ. Действительно, при решении этой же задачи процедура `ode23` выполнила 315

шагов, вычислив функцию f 946 раз. Однако следует иметь в виду, что процедура `ode15s` работает медленнее, т.к. в соответствии с ее алгоритмом необходимо решать множество систем из 128 линейных уравнений. В MATLAB решение подобных уравнений реализовано очень эффективно, однако накладные расходы, связанные с использованием численных процедур, основанных на МДН, делают в рассматриваемом случае процедуру Рунге–Кутта `ode45` более эффективной. С другой стороны, явные методы Рунге–Кутта высокого порядка менее эффективны по сравнению с `ode15s`. Поскольку формулы, на которых основаны эти методы, обладают меньшей областью устойчивости, процедура `ode45` менее эффективна для рассматриваемого примера, чем `ode23`.

ПРИМЕР 2.3.9

Рассмотренные в примере 2.3.8 спектральные аппроксимации u_x могут быть вычислены очень точно, если функция $u(x,t)$ является гладкой. Альтернативным подходом является использование конечно-разностных аппроксимаций. Эти аппроксимации имеют (существенно) меньший порядок точности, но они являются локальными по своей природе и поэтому могут быть использованы при получении решений, о которых известно, что они (существенно) менее гладкие. Важно подчеркнуть, что одностороннее волновое уравнение может иметь решения с пространственными переменными, терпящими разрывы, и это обстоятельство требует особого внимания при численном решении соответствующих ДУЧП. В качестве простого примера решения этого волнового уравнения с использованием схемы с конечными разностями против потока можно рассмотреть разностную аппроксимацию первого порядка для производной пространственной переменной. Как и ранее, мы будем использовать сетку по x, состоящую из N точек, равноразнесенных с шагом h. Поскольку $c(x) > 0$, волна движется направо и в результате применения конечно-разностной схемы мы получаем

$$\frac{dv_m}{dt} = -c(x_m)\left(\frac{v_m - v_{m-1}}{h}\right).$$

Здесь и далее мы будем предполагать, что задача рассматривается при периодических граничных условиях, поскольку в этом случае использование спектральных аппроксимаций упрощается. С учетом этого предположения на левой границе имеем

$$\frac{dv_1}{dt} = -c(x_1)\left(\frac{v_1 - v_N}{h}\right). \tag{2.38}$$

Мы вернемся к рассмотрению этого случая позднее, но в настоящий момент предлагаем изменить граничные условия так, чтобы еще более упростить применение конечно-разностной схемы. При заданных начальных условиях в качестве граничного условия удобно рассмотреть $u_x(0,t) = 0$, аппроксимация которого удовлетворяет $v_1'(t) = 0$. При использовании аппроксимации первого порядка для $u_x(x,t)$ для получения численного решения, сравнимого с изображенным на рисунке 2.7, нам необходимо иметь достаточно частую сетку. В программе `ch2ex6.m` эта цель достигается при $N = 1000$. Заметим, что мы не выводим значения решения во всех точках столь частой сетки.

Представленная программа, использующая конечно-разностную схему, довольно проста, но при решении задач с большим числом уравнений (в рассматриваемом здесь примере — 1000) возникают новые проблемы. Задачи, решаемые с использованием метода прямых с достаточно частой дискретизацией по пространственным переменным, могут быть жесткими. Действительно, параболические ДУЧП, при решении которых используется процедура `pdepe`, являются жесткими при достаточно частой пространственной сетке. Рассматриваемая ЗНУ не является жесткой, но она достаточно проста, что позволит нам выработать на этом примере стратегию решения проблем, связанных с жесткостью системы. В численных процедурах, предназначенных для решения жестких задач, вычисляются матрицы Якоби, размерность которых при N уравнениях равна $N \times N$. В контексте рассматриваемых в этом параграфе задач приходится решать системы из нескольких тысяч уравнений и поэтому важное значение имеет фактор разреженности матрицы Якоби, при учете которого можно снизить вычислительные затраты при решении соответствующих систем линейных уравнений. Технологии обработки разреженных матриц являются неотъемлемой частью среды программирования MATLAB, но в других численных процедурах, предназначенных для выполнения научных расчетов, эта возможность может быть не реализована. Действительно, насколько нам известно, в некоторых подобных процедурах имеется возможность учесть лишь тот факт, что матрица Якоби является *ленточной*. (Ленточная матрица — это матрица, все ненулевые элементы которой заключены внутри ленты, образованной диагоналями, параллельными главной диагонали.) Необходимо каким-либо образом проинформировать численную процедуру о наличии нулевых элементов в матрице Якоби $J = \left(\frac{\partial f_m}{\partial v_k}\right)$. Наиболее простой способ — это представить разреженную матрицу Якоби аналитически, что может быть сделано с использованием опции `Jacobian`. Если матрица Якоби является постоянной, в качестве значения этой опции следует использовать значение этой матрицы. Соответствующий текст в программе `ch2ex6.m` имеет следующий вид

```
B = [ [-c_h(2:N); 0] [0; c_h(2:N)] ];
J = spdiags(B,-1:0,N,N);
options = odeset('Jacobian',J);
```

Если матрица Якоби не является постоянной, для вычисления ее значения необходимо написать соответствующую функцию, выходным значением которой является разреженная матрица. В любом случае численная процедура должна быть проинформирована о том, что матрица Якоби является разреженной. Если не представляет большого труда представить матрицу Якоби в аналитическом виде, это всегда должно быть сделано, поскольку с вычислительной точки зрения аппроксимация значения матрицы Якоби представляет собой высокозатратную операцию. Кроме того, если матрица Якоби задана аналитически, численные процедуры становятся еще более эффективными. В численных процедурах MATLAB по умолчанию выполняется численная аппроксимация матриц Якоби, поскольку этот вариант удобен для пользователей. Тем не менее при решении сложных задач очень важно проинформировать численную процедуру о структуре матрицы Якоби. Элемент матрицы Якоби $J_{m,k}$ тождественно равен нулю, если m-ое уравнение не зависит от k-ой

компоненты решения. Для того, чтобы ввести в численную процедуру информацию о разреженности матрицы Якоби необходимо создать матрицу S и назначить ее элементам значения ноль и единица, причем нули должны быть расположены там, где соответствующие элементы матрицы Якоби тождественно равны нулю, а единицы — там, где соответствующие элементы матрицы Якоби не равны нулю. Матрица S определяет структуру разреженной матрицы Якоби и эта информация вводится в численную процедуру с использованием опции `JPattern` со значением S (см. текст программы `ch2ex6.m`). В рассматриваемом здесь случае ненулевые элементы расположены на диагоналях и поэтому при инициализации разреженной матрицы удобно воспользоваться функцией `spdiags`. В программе `ch2ex6.m` это могло бы быть сделано следующим образом

```
S = spdiags(ones(N,2),-1:0,N,N);
S(1,1) = 0;
```

В упражнении 2.34 читателю предлагается показать, что рассматриваемая здесь ЗНУ является нежесткой. Однако в целях иллюстрации решения жестких систем большой размерности численное решение этой задачи выполняется с использованием `ode15s` (см. текст программы `ch2ex6.m`). Матрица Якоби аппроксимируется с использованием конечно-разностной схемы и в целях эффективности ее применения задается структура этой разреженной матрицы. Следует отметить, что для увеличения эффективности вычислений с использованием этой постоянной матрицы следует написать специальную функцию, реализующую соответствующее аналитическое представление и передаваемую в численную процедуру с использованием опции `Jacobian`.

```
function ch2ex6
global c_h

% Spatial grid, initial values:
N = 1000;
h = 2*pi/N;
x = h*(1:N);
v0 = exp(-100*(x - 1) .^ 2);
c_h = - (0.2 + sin(x - 1) .^ 2)' / h;

% Sparsity pattern for the Jacobian:
S = sparse(N,N);
for m = 2:N
    S(m,m-1) = 1;
    S(m,m) = 1;
end
options = odeset('JPattern',S);

% Integrate and plot:
t = 0:0.30:8;
[t,v] = ode15s(@f,t,v0,options);
pltspace = ceil(N/128);
x = x(1:pltspace:end);
```

```
v = v(:,1:pltspace:end);
surf(x,t,v), view(10,70), axis([0 2*pi 0 8 0 5])
ylabel t, zlabel u, grid off

%=============================================
function dvdt = f(t,v)
global c_h
dvdt = c_h .* [0; diff(v)];
```

При периодических граничных условиях в рассматриваемом примере матрица Якоби не ограничена, поскольку первое уравнение зависит от v_N. По этой причине стандартные численные процедуры не позволяют непосредственно решить эту ЗНУ. В отличие от них, предназначенные для решения жестких задач численные процедуры MATLAB позволяют это сделать: для этого необходимо лишь вставить после декларации матрицы S следующие два выражения

```
S(1,1) = 1;
S(1,N) = 1;
```

В упражнении 2.35 читателю предлагается решить эту ЗНУ с периодическими граничными условиями.

Если проигнорировать разреженность матрицы Якоби, вычисление ее аппроксимации с использованием конечно-разностной схемы сопряжено по крайней мере с одним вычислением функции $f(t,v)$ для каждого из N столбцов этой матрицы. В численных процедурах MATLAB используется предложенный в работе [Curtis et al., 1974] алгоритм, который позволяет существенно снизить соответствующие вычислительные затраты с учетом информации о наличии в матрице Якоби заведомо нулевых элементов. Например, если матрица Якоби является ленточной и имеет D диагоналей, то этот алгоритм позволяет вычислить эту матрицу, выполнив лишь D вычислений функции $f(t,v)$, т. е. *количество этих вычислений не зависит от размерности системы* N. В рассмотренной выше программе `ch2ex6.m`, при аппроксимации матрицы Якоби с использованием информации о структуре матрицы Якоби, функцию $f(t,v)$ потребовалось вычислить лишь *дважды*; если же информацию о разреженности матрицы Якоби проигнорировать, то функцию $f(t,v)$ потребовалось бы вычислить $D = N = 1000$ раз. Из этого примера становится очевидно, что учет разреженности матрицы Якоби имеет очень важное значение для увеличения эффективности численного решения линейной системы с итерационной матрицей, а также с точки зрения минимизации используемого объема оперативной памяти компьютера.

ПРИМЕР 2.3.10

Как мы видели в предыдущем примере, одним из путей увеличения скорости вычисления численной аппроксимации матрицы Якоби является использование информации о разреженности матрицы Якоби для уменьшения количества вычислений функции $f(t,y)$ в правой части рассматриваемого дифференциального уравнения. Другой подход основан на увеличении скорости самих вычислений и эта цель может быть достигнута с использованием их векторизации. Очень часто векторизация существующих программ MATLAB может быть выполнена без особого труда.

При аппроксимации матрицы Якоби функция $f(t,y)$ вычисляется при нескольких различных значениях y, но при одном значении t. Это обстоятельство может быть использовано при написании программного кода соответствующей процедуры: если она вызывается с аргументами `(t,[y1 y2 ... ])`, то после выполнения соответствующих вычислений она должна возвращать `[f(t,y1) f(t,y2) ... ]`. Для того, чтобы проинформировать об этом численную процедуру, необходимо использовать опцию `Vectorized` со значением `on`. Поскольку все встроенные функции MATLAB векторизованы подобным образом, векторизация процедуры вычисления $f(t,y)$ часто сводится к реализации `y` в виде массива вектор-столбцов и использованию точки перед операторами, что указывает на то, что они применяются к соответствующим массивам покомпонентно. Модель брюсселятора представляет собой систему из двух ДУЧП. Решение этой системы было получено в работе [Hairer & Wanner, 1991, стр. 6–8] с использованием метода прямых с конечно-разностной дискретизацией по пространственной переменной. Численное решение этой задачи представлено в виде одной из демонстрационных программ MATLAB. При запуске программы `brussode(N)` формируется и решается соответствующая система из $2N$ обыкновенных дифференциальных уравнений. Принятое по умолчанию значение параметра N равно 20. Здесь мы рассмотрим только одну строку из текста этой программы, в которой выполняется векторизация вычисления функции $f(t,y)$. Некоторые из компонент вектора `dydt` вычисляются в теле цикла по `i`

```
dydt(i) = 1 + y(i+1)*y(i)^2 - 4*y(i) + ...
              c*(y(i-2) - 2*y(i) + y(i+2));
```

В процессе этих вычислений используются значения компонент вектор-столбца `y(:)`. Векторная версия представленного выше отрывка программного кода должна выполнять те же операции над столбцами массива `y(:,:)`. В программе `brussode` это реализовано следующим образом

```
dydt(i,:) = 1 + y(i+1,:).*y(i,:).^2 - 4*y(i,:) + ...
                c*(y(i-2,:) - 2*y(i,:) + y(i+2,:));
```

Точка перед операторами умножения и вычисления степени указывает на то, что эти операторы должны быть применены покомпонентно к элементам соответствующих массивов. Например, прибавление 1 к массиву интерпретируется как прибавление единицы к каждой компоненте этого массива. Следует отметить, что в рассматриваемом случае эффект векторизации вычислений будет малозаметен. Структура разреженной матрицы Якоби такова, что необходимо вычислить лишь несколько значений этой матрицы и при каждом таком вычислении необходимо несколько раз вычислить значение функции $f(t,y)$. Действительно, используя опцию `Stats` со значением `on` можно убедиться в том, что в процессе интегрирования системы было сформировано лишь две матрицы Якоби. Значимость применения векторизации зависит от структуры разреженной матрицы Якоби, а также от вычислительной трудоемкости вычисления функции $f(t,y)$. Аналогичная векторизация программного кода может оказаться полезной при решении ЗГУ с использованием `bvp4c`, и мы вернемся к этому вопросу в главе 3.

В некоторых случаях бывает неясно, как выполнить векторизацию программного кода. В качестве иллюстрации подобной ситуации рассмотрим программу ch2ex5.m, заменим в ее тексте вызов процедуры ode23 на вызов процедуры ode15s и попытаемся выполнить векторизацию следующих операторов

```
dvdt = mc .* real(ifft(ixindices .* fft(v)));
```

Процедуры вычисления быстрого преобразования Фурье векторизованы, поэтому fft(v) будет вычисляться векторно, если массив v содержит более чем один столбец. Проблемы возникают при формировании ixindices .* fft(v). Действительно, ixindices является вектор-столбцом и его размерность не соответствует размерности столбцов fft(v), что необходимо для покомпонентного умножения. Эту операцию можно выполнить в цикле. При этом следует принять во внимание тот факт, что численная процедура выполняет вызов функции с аргументом v, являющимся в общем случае массивом с N столбцами, где N иногда может быть равно единице. Для более эффективного вычисления производных следует сделать ixindices матрицей с N идентичными столбцами. Это может быть выполнено с использованием функции repmat

```
temp1 = i * [0:N/2-1 0 -N/2+1:-1]';
ixindices = repmat(temp1,1,N);
```

При выполнении операции умножения массивов ixindices и fft(v) следует обеспечить соответствие их размерностей. Для этого необходимо определить число столбцов матрицы v, воспользовавшись функцией size, после чего при умножении использовать соответствующее число столбцов матрицы ixindices. Аналогичные действия должны быть выполнены и в отношении массива mc. Тогда вычисление dvdt может быть выполнено с использованием следующего программного кода

```
nc = size(v,2);
dvdt = mc(:,1:nc) .* real(ifft(ixindices(:,1:nc) .* fft(v)));
```

При решении рассматриваемой в этом примере ЗНУ векторизация вычисления f(t,v) приводит к незначительному увеличению скорости работы программы, поскольку эта задача является нежесткой и в численной процедуре ode15s матрица Якоби вычисляется лишь один раз. Однако основной нашей целью при рассмотрении этого примера была иллюстрация проблем, которые могут возникать при векторизации программного кода.

ПРИМЕР 2.3.11

Дифференциальные уравнения в форме представления с матрицей весовых коэффициентов возникают при выполнении пространственной дискретизации с использованием метода Галеркина. В целях иллюстрации этого метода, а также в контексте обсуждения вопросов, связанных с исследованием больших жестких систем, рассмотрим уравнение теплопроводности, исследованное в работе [Fletcher, 1984]

$$u_t = u_{xx},\ 0 \leqslant x \leqslant 1$$

с начальным условием

$$u(x,0) = x + \sin(\pi x)$$

и граничными условиями

$$u(0,t) = 0,\ u(1,t) = 1.$$

Приближенное решение ищется в виде

$$v(x,t) = \sum_m S_m(x) v_m(t).$$

При заданном значении t, эта аппроксимация удовлетворяет ДУЧП с невязкой

$$R(x,t) = v_t(x,t) - v_{xx}(x,t).$$

Скалярное произведение двух непрерывных на $[0,1]$ функций определяется равенством

$$(f,g) = \int_0^1 f(x)g(x)dx.$$

В соответствии с методом Галеркина $u(x,t)$ проектируется на пространство функций вида $v(x,t)$ путем наложения требования ортогональности невязки аппроксимации базисным функциям $S_k(x)$, т. е.

$$(R, S_k) = (v_t, S_k) - (v_{xx}, S_k) = 0$$

для всех k. Если каждая функция формы (shape function) $S_m(x)$ является кусочно-линейной функцией, удовлетворяющей $S_m(x_m) = 1$ и $S_m(x_j) = 0$ при $j \neq m$, то

$$v(x_m,t) = v_m(t) \approx u(x_m,t).$$

Учитывая вид $v(x,t)$ и выполняя интегрирование по частям, получаем уравнение

$$\sum_m (S_m, S_k)\frac{dv_m}{dt} + \sum_m \left(\frac{dS_m}{dx}, \frac{dS_k}{dx}\right) v_m = 0.$$

В случае, когда шаг сетки h постоянен, вычисляя скалярное произведение, получаем систему ОДУ

$$\frac{1}{6}\frac{dv_{m-1}}{dt} + \frac{4}{6}\frac{dv_m}{dt} + \frac{1}{6}\frac{dv_{m+1}}{dt} = \frac{v_{m-1} - 2v_m + v_{m+1}}{h^2},$$

где $m = 1, 2, \ldots, N$. С учетом граничных условий получаем

$$v_0(t) = 0,\ v_{N+1}(t) = 1.$$

Эти величины присутствуют в вышеприведенной системе ОДУ только в первом и последнем уравнениях, причем это последнее уравнение является неоднородным. Из начального условия следует, что $v_m(0) = u(x_m, 0)$ при $m = 1, 2, \ldots, N$.

Множество различных задач могут быть сформулированы с использованием системы ОДУ с матрицей весовых коэффициентов. Если эта матрица $M(t,y)$ не сильно зависит от y, для решения соответствующей задачи можно легко модифицировать численные методы так, чтобы учесть наличие этой матрицы. При численном исследовании больших систем важно принимать во внимание структуру

матрицы весовых коэффициентов, а также структуру матрицы Якоби. Несмотря на то, что форма представления исследуемой системы ОДУ именно в виде (2.37) часто бывает наиболее удобной и эффективной, стандартные численные процедуры, основанные на МДН, не допускают соответствующего формата входных данных. Это обусловлено тем, что эти программы обладают недостаточно развитым интерфейсом с пользователем, а также связано с ограничениями на допустимый объем оперативной памяти используемого компьютера. Например, если матрица весовых коэффициентов постоянна, при модификации методов МДН следует в качестве итерационной матрицы использовать не $I - h\gamma J$, как в стандартном случае, а $M - h\gamma J$. Эта модификация приводит к необходимости выполнения соответствующих изменений программного кода для декларации и хранения в памяти компьютера соответствующих матриц, поскольку особенности структуры M и J должны быть использованы при формировании итерационной матрицы. Все эти проблемы отсутствуют при использовании численных процедур MATLAB для решения ЗНУ, т. к. все эти матрицы представлены в виде разреженных матриц общего вида, а сама система MATLAB обеспечивает автоматическое решение проблем, связанных с выделением необходимого объема оперативной памяти.

Характер необходимых модификаций численной процедуры, обусловленных использованием матрицы весовых коэффициентов $M(t, y)$, определяется степенью зависимости этой матрицы от y. Чем меньше эта зависимость, тем легче выполнить соответствующую модификацию метода. Наиболее простая ситуация возникает, когда матрица весовых коэффициентов постоянна. В этом случае численная процедура должна вызываться с опцией `Mass` со значением, равным соответствующей матрице. Если матрица весовых коэффициентов не является постоянной, используется опция `MStateDependence`, позволяющая проинформировать численную процедуру о том, насколько сильно матрица $M(t, y)$ зависит от y. Эту опцию разумно использовать лишь при рассмотрении систем достаточно большой размерности. В качестве значений этой опции могут быть использованы следующие. Одно из возможных значений `none` соответствует ситуации, когда матрица весовых коэффициентов не зависит от y, т. е. $M(t, y) = M(t)$. Значением, принятым по умолчанию, является `weak`. С теоретической и практической точек зрения наиболее сложной ситуации соответствует значение `strong`. Если $M(t, y)$ сильно зависит от y, решение дифференциального уравнения в форме (2.37) представляет собой непростую задачу и в качестве вычислительных алгоритмов, пригодных для ее решения, следует рассматривать те, что предназначены для решения систем ОДУ общего вида

$$F(t, y, y') = 0. \tag{2.39}$$

Эффективное решение больших систем ОДУ с матрицей весовых коэффициентов, сильно зависящей от переменной состояния, связано со значительными трудностями и поэтому в качестве значения опции `MStateDependence` мы рекомендуем использовать `weak`. Этот случай хорошо описан в документации MATLAB и в большинстве случаев при этом значении могут быть получены вполне удовлетворительные результаты.

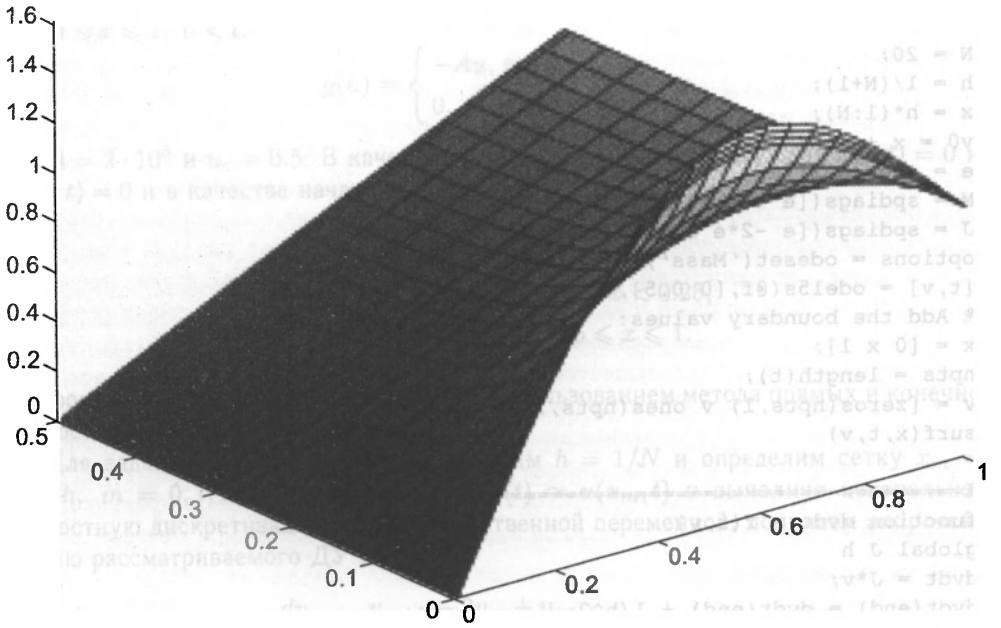

Рис. 2.8. Решение дифференциального уравнения $u_t = u_{xx}$.

Существует несколько прикладных программ, допускающих решение задач, представленных в виде (2.39) (см., например, DASSL [Brenan et al., 1996]). Несомненно, будущие версии MATLAB будут включать численные процедуры для решения подобных задач. Этому классу систем принадлежат алгебро-дифференциальные уравнения (АДУ). С теоретической и вычислительной точек зрения исследование уравнений общего вида (2.39) является существенно более сложной задачей, чем решение систем ОДУ. Очевидной теоретической проблемой является то обстоятельство, что в общем случае задание начального значения $y(t_0)$ недостаточно для определения решения — дополнительно необходимо задать значение производной в начальный момент времени $y'(t_0)$, причем это значение должно быть непротиворечиво в том смысле, что при заданных t_0, $y(t_0)$ и $y'(t_0)$ должно быть выполнено уравнение (2.39). Кроме того совершенно очевидно, что в численной процедуре, предназначенной для решения подобных уравнений, должны вычисляться $\frac{\partial F}{\partial y'}$ и $\frac{\partial F}{\partial y}$, что усложняет ее интерфейс с пользователем и управление выделенным объемом оперативной памяти.

Поскольку для рассматриваемого в этом параграфе примера матрица весовых коэффициентов и матрица Якоби являются постоянными, их удобно передать в численную процедуру непосредственно (по значению). На рисунке 2.8 показаны результаты работы программы `ch2ex7.m`. Для лучшего понимания качественных свойств полученного решения и выбора наиболее подходящей проекции удобно воспользоваться средством вращения графика `Rotate 3D`.

```
function ch2ex7
global J h
```

```
N = 20;
h = 1/(N+1);
x = h*(1:N);
v0 = x + sin(pi*x);
e = ones(N,1);
M = spdiags([e 4*e e],-1:1,N,N)/6;
J = spdiags([e -2*e e],-1:1,N,N)/h^2;
options = odeset('Mass',M,'Jacobian',J);
[t,v] = ode15s(@f,[0 0.5],v0,options);
% Add the boundary values:
x = [0 x 1];
npts = length(t);
v = [zeros(npts,1) v ones(npts,1)];
surf(x,t,v)

%=======================================
function dvdt = f(t,v)
global J h
dvdt = J*v;
dvdt(end) = dvdt(end) + 1/h^2;
```

■ УПРАЖНЕНИЕ 2.34

Модифицируйте программу `ch2ex6.m` так, чтобы в ней использовалась численная процедура `ode23`. С использованием функций `tic` и `toc` определите как эта модификация повлияла на время выполнения программы и сделайте соответствующий вывод о жесткости рассматриваемой ЗНУ. Эта задача не может быть жесткой, если вы сможете получить ее решение с использованием `ode23`.

■ УПРАЖНЕНИЕ 2.35

В примере 2.3.9 было указано, что случай периодических граничных условий является наиболее простым, поскольку реализация `ode15s` допускает использование разреженных матриц общего вида. Убедитесь в этом, модифицировав программу `ch2ex6.m` так, чтобы с ее помощью можно было решить задачу с периодическими граничными условиями. Матрицу S, определяющую структуру разреженной матрицы Якоби, следует сформировать так же, как это было сделано в упомянутом примере. Для решения задачи в случае периодического граничного условия (2.38) необходимо изменить лишь первую компоненту `dvdt`.

■ УПРАЖНЕНИЕ 2.36

Модель фронта быстрого охлаждения раскаленного металлического стержня при его погружении в холодную жидкость имеет вид дифференциального уравнения в частных производных (ДУЧП) [Laquer & Wendroff, 1981]

$$u_t = u_{xx} + g(u),$$

при $0 \leqslant x \leqslant 1,\ 0 < t,$

$$g(u) = \begin{cases} -Au, & \text{если } u \leqslant u_c, \\ 0, & \text{если } u_c < u, \end{cases}$$

где $A = 2 \cdot 10^5$ и $u_c = 0.5$. В качестве граничных условий выступают $u(0,t) = 0$ и $u_x(1,t) = 0$ и в качестве начального условия

$$u(x,0) = \begin{cases} -0, & \text{если } 0 \leqslant x \leqslant 0.1, \\ \frac{x-0.1}{0.15}, & \text{если } 0.1 < x < 0.25, \\ -1, & \text{если } 0.25 \leqslant x \leqslant 1. \end{cases}$$

Аппроксимируем решение этого ДУЧП с использованием метода прямых и конечно-разностной схемы.

Для заданного целого числа N положим $h = 1/N$ и определим сетку $x_m = mh$, $m = 0, 1, \ldots, N+1$. Полагая $v_m(t) \approx u(x_m, t)$ и выполняя центрально-разностную дискретизацию по пространственной переменной, получаем аппроксимацию рассматриваемого ДУЧП

$$\frac{dv_m}{dt} = \frac{v_{m+1} - 2v_m + v_{m-1}}{h^2} + g(v_m).$$

Легко видеть, что для определения решения первого уравнения ($m = 1$) необходимо задать $v_0(t)$. С учетом граничного условия $u(0,t) = 0$ определим $v_0(t) = 0$ и исключим эту величину из уравнения. Аналогично, для определения решения последнего уравнения ($m = N$) необходимо задать $v_{N+1}(t)$. Аппроксимируем граничное условие в $x = 1$ с использованием центральной разности

$$0 = u_x(1,t) \approx \frac{v_{N+1} - v_{N-1}}{2h}.$$

Тогда можно положить $v_{N+1} = v_{N-1}$ и исключить эту величину из уравнения для $m = N$. Начальные условия для вышеприведенных ОДУ определяются равенствами $v_m(0) = u(x_m, 0)$. Найдите численное решение этой системы уравнений $m = 1, 2, \ldots, N$ при $N = 50$ с учетом того, что матрица Якоби J является постоянной и разреженной матрицей. Эта задача может быть решена аналогично тому, как это было сделано в программе `ch2ex7.m`, но в рассматриваемом здесь случае $J_{N,N-1} = 2h^{-2}$. Полезно также векторизовать вычисление $g(v_m)$. Для этого следует ответить на вопрос, что означает запись `(w <= 0,2)`, если `w` является вектор-столбцом. Вывод данных выполните при `tspan =0:0.001:0.006`. При графическом выводе вычисленного решения `v` для различных значений пространственной сетки по `x` вы получите эволюцию профилей решения с течением времени. Решите эту ЗНУ с использованием `ode15s` и `ode23`. Используйте функции `tic` и `toc` для определения вычислительной трудоемкости в каждом из этих случаев. Убедитесь в том, что эта ЗНУ является нежесткой, т. к. `ode23` оказывается более эффективной по сравнению с `ode15s`. Исследователи, хорошо знакомые с использованием метода прямых, возможно будут удивлены, узнав об этом результате. Для того, чтобы понять причину их удивления, решите рассматриваемую задачу при нулевом члене

$g(u)$ на промежутке `tspan = 0:0.1:0.6`. Убедитесь в том, что эта ЗНУ является (умеренно) жесткой.

Для объяснения этой ситуации заметим, что член $g(u)$ является негладким. Это оказывает влияние на компоненты решения $v_m(t)$, которые демонстрируют разрыв при прохождении решения через значение u_c. В рассматриваемой задаче это случается один раз для каждой из m компонент. Численная процедура должна локализовать эту точку и использовать при ее прохождении меньший шаг. В общем случае одна изолированная точка, в которой решение терпит разрыв, в целом не сильно влияет на процесс интегрирования, но в рассматриваемой задаче это событие случается в различные моменты времени для различных компонент решения. С учетом того, что этих компонент может быть достаточно много, совокупный объем вычислений может существенно возрасти. Однако, как вы помните, ЗНУ является жесткой, если решение является достаточно гладким, но шаг интегрирования определяется требованиями устойчивости используемого явного метода. В рассматриваемом случае эти требования никак не ограничивают шаг интегрирования и поэтому численная процедура `ode15s` оказывается менее эффективной по сравнению с `ode23` даже при заданной аналитически и постоянной матрице Якоби.

2.3.4. Сингулярности

При реализации стандартных численных процедур интегрирования систем ОДУ обычно предполагается, что решение в определенной степени является гладким и при невыполнении этого требования в изолированной точке необходимо выполнить некоторую работу аналитического характера. Основной подход, используемый в подобных ситуациях, заключается в том, что вблизи сингулярной точки интересующее нас решение аппроксимируется рядом или асимптотическим разложением, а на остальном интервале интегрирования это решение аппроксимируется с использованием стандартной численной процедуры решения ЗНУ. Выражение «интересующее нас решение» неявно указывает на то обстоятельство, что в сингулярной точке может существовать более одного решения. Сингулярности часто возникают при рассмотрении задач с граничными условиями (ЗГУ) и это обусловлено тем, что эти задачи часто ставятся на бесконечном интервале. Более подробно эти вопросы будут рассмотрены в главе 3, а в этом параграфе мы познакомимся с общей схемой решения таких задач. Для этого мы рассмотрим один пример и предложим читателю выполнить пару упражнений.

ПРИМЕР 2.3.12

Классические методы анализа сжатия сферической полости в жидкости [см. Lighthill, 1986, стр. 103ff] приводят к рассмотрению следующей ЗНУ

$$(y')^2 = \tfrac{2}{3}(y^{-3} - 1),\ y(0) = 1. \tag{2.40}$$

Переменные x (время) и y (радиус полости) являются безразмерными. Процесс интегрирования должен быть остановлен в момент, когда значение переменной y становится равным нулю, что соответствует полному коллапсу полости. Уравнение (2.40) может быть представлено в стандартной форме

$$\frac{dy}{dx} = -\sqrt{\tfrac{2}{3}(y^{-3} - 1)}.$$

Выбор знака перед квадратным корнем (минус) обусловлен тем, что это уравнение моделирует процесс *сжатия (коллапса)* полости. При решении этой ЗНУ возникают две трудности. Для существования и единственности решения ЗНУ необходимо, чтобы функция f в правой части соответствующего ОДУ $y' = f(x, y)$ была гладкой в области, содержащей начальную точку $(0, 1)$. Однако в рассматриваемом случае это требование не выполнено, т. к. начальная точка расположена на границе области, где определена функция f. Таким образом, возможно, существует более чем одно решение. Действительно, наряду с искомым, физически обусловленным убывающим решением, существует тривиальное решение $y(x) \equiv 1$. Поэтому необходимо выработать стратегию выбора «правильного» решения. Другая трудность заключается в том, что в момент полного коллапса полости x_c мы имеем $y(x_c) = 0$ и из ОДУ видно, что $y'(x_c) = -\infty$.

Для решения первой проблемы аппроксимируем решение $y(x)$ при $0 \leqslant x \leqslant d$ рядом Тейлора и используем вычисленное с использованием этого ряда решение $y_d \approx y(d)$ на начальном этапе численного интегрирования при $x \geqslant d$, где ОДУ не является сингулярным. Имеющийся в MATLAB пакет символьных вычислений позволяет легко получить выражение для ряда Тейлора. С учетом заданного начального значения искомая аппроксимация должна иметь вид $y(x) = 1 + ax + bx^2 + \ldots$ Нижеследующие команды позволяют подставить полученное выражение в рассматриваемое ОДУ

```
syms y x a b res
y = 1 + a*x + b*x^2;
res = taylor(diff(y)^2 - (2/3)*(1/y^3 - 1),3)
```

Далее вычисляются первые три члена соответствующей невязки

```
res = a^2+(4*a*b+2*a)*x+(4*b^2+2*b-4*a^2)*x^2
```

Для того, чтобы обеспечить наилучшую аппроксимацию ОДУ, коэффициенты в разложении решения необходимо выбрать так, чтобы начиная с члена наименьшего порядка, максимальное количество членов в выражении для невязки стало равным нулю. Для исключения из этого выражения постоянного члена положим $a = 0$. При этом в выражении для решения исключается член первого порядка. Член невязки при x^2 может быть исключен при двух вариантах выбора b: $b = 0$ или $b = -0.5$. Таким образом, эта сингулярная ЗНУ имеет два решения, которые могут быть разложены в соответствующие ряды Тейлора. Одним из этих решений является $y(x) \equiv 1$, существование которого было предсказано нами ранее. Другое решение имеет вид

$$y(x) = 1 - \tfrac{1}{2}x^2 + \ldots$$

и оно является убывающим при малых x. Только это решение обладает физически мотивированными свойствами и поэтому оно является для нас искомым. Продолжая процесс определения коэффициентов разложения в выражении для решения, окончательно получаем

$$y(x) = 1 - \tfrac{1}{2}x^2 - \tfrac{1}{6}x^4 - \tfrac{19}{180}x^6 + \ldots$$

В программе ch2ex8.m мы использовали три члена в выражении для разложения $y(x)$ и оценили величину ошибки этой аппроксимации с использованием следующего члена этого разложения. Относительная ошибка этой аппроксимации на промежутке до d оказалась приблизительно равной 10^{-7} при $d = 0.1$.

Трудности вычисления решения в конце интервала интегрирования, связанные с тем, что производная принимает бесконечное значение, могут быть решены путем взаимной замены независимой и зависимой переменных. Этот метод широко применяется при аналитическом исследовании ОДУ и очень полезен при нахождении численных решений. При использовании этого подхода необходимо быть внимательным и убедиться в допустимости подобной замены в исследуемой задаче. В нашем примере из ОДУ видно, что $y(x)$ является строго убывающей функцией и поэтому переменную y можно использовать в качестве независимой, если решение достаточно удалено от начальной точки. Действительно, в начальной точке переменная y не может служить в качестве независимой переменной, т. к. $y'(x)$ обращается в ноль при $x = 0$ и снова возникает та же проблема, связанная с бесконечным значением производной. С учетом всех этих обстоятельств для решения задачи следует выполнить интегрирование ОДУ

$$\frac{dx}{dy} = -\sqrt{\frac{3y^3}{2(1-y^3)}}$$

с начальным значением $x = d$ при y, изменяющейся от $y = y_d$ до $y = 0$. Длительность интервала времени, необходимого для полного коллапса полости, равна значению x при $y = 0$. После выполнения этой предварительной аналитической работы задача может быть численно решена с использованием программы ch2ex8.m. Для того, чтобы вывести график решения на всем интервале интегрирования, нам необходимо объединить данные, полученные путем численного интегрирования указанного дифференциального уравнения, с данными, полученными в результате вычисления значения аппроксимирующего ряда на интервале $[0, d]$. В рассматриваемом случае достаточно гладкий график может быть получен, если значение ряда вычислено только в одной точке — при начальном значении $y(0) = 1$. Читателю предлагается выполнить эксперименты, меняя значение величины d на большее или меньшее чем $d = 0.1$.

```
function ch2ex8
d = 0.1;
yd = 1 - (1/2)*d^2 - (1/6)*d^4;
erryd = (19/180)*d^6;
fprintf('At d = %g, the error in y(d) is about %5.1e.\n',d,erryd);

[y,x] = ode45(@ode,[yd 0],d);
fprintf('Total collapse occurs at x = %g.\n',x(end));

% Augment the arrays with y = 1 at x = 0 for the plot
% and plot with the original independent variable x.
y = [1; y];
x = [0; x];
plot(x,y)
```

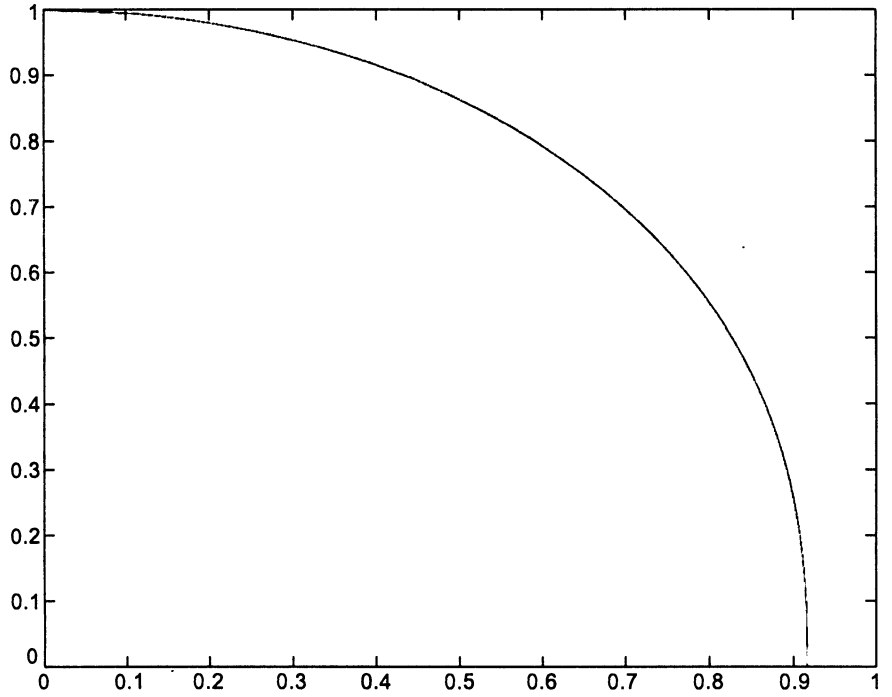

Рис. 2.9. Решение задачи о коллапсе полости.

```
%=============================================================
function dxdy = ode(y,x)
dxdy = - sqrt(3*y^3/(2*(1 - y^3)));
```

Результатами работы этой программы являются график, изображенный на рисунке 2.9, и следующий текст, выводимый на экран

```
>> ch2ex8
At d = 0.1, the error in y(d) is about 1.1e-007.
Total collapse occurs at x = 0.914704
```

■ УПРАЖНЕНИЕ 2.37

В работе [Davis, 1962, стр. 371ff] в качестве тепловой модели облака газа сферической формы рассматривается уравнение Эмдена

$$\frac{d^2y}{dx^2} + \frac{2}{x}\frac{dy}{dx} + y^n = 0,$$

где n — некоторый физический параметр. Сингулярный коэффициент возникает, когда с учетом сферической симметрии это ДУЧП сводится к ОДУ. Из свойства симметрии следует, что начальное значение удовлетворяет $y'(0) = 0$ и в соответствующих безразмерных переменных $y(0) = 1$. Из физических соображений можно

заключить, что решение является ограниченным и гладким в начале координат и, следовательно, может быть разложено в ряд Тейлора. В указанной работе для решения было получено следующее разложение в ряд

$$y(x) = 1 - \frac{x^2}{3!} + n\frac{x^4}{5!} + (5n - 8n^2)\frac{x^6}{3 \cdot 7!} + \ldots$$

Если рассматривать эту модель в качестве модели звезд, то первое нулевое значение решения соответствует радиусу звезды. При моделировании более яркой компоненты двойной системы Капелла[1] использовалось значение параметра $n = 3$. При этом было установлено, что первый нуль решения возникает вблизи точки $x = 6.9$. Читателю предлагается проверить правильность этого результата. Используя первые три члена в разложении решения, вычислите приближенное значение $y(0.1)$; с использованием выражения для производной этого ряда, вычислите приближенное значение $y'(0.1)$. Оцените ошибку этих аппроксимаций. Используя полученные аналитические значения аппроксимаций величин $y(0.1)$ и $y'(0.1)$ в качестве начальных значений, а также численную процедуру `ode45`, выполните интегрирование рассматриваемого ОДУ до возникновения терминального события $y(x) = 0$. Вам необходимо самостоятельно определить необходимую длительность интервала интегрирования и при прерывании работы программы убедитесь в том, что это прерывание произошло именно вследствие возникновения терминального события. Для того чтобы быть уверенным в точной локализации первого нуля решения $y(x)$, аппроксимация решения при $x = 0.1$ должна быть вычислена очень точно. Поэтому при интегрировании следует использовать отличные от принятых по умолчанию значения допустимых ошибок вычислений: например, в качестве допустимых значений относительной и абсолютной ошибок можно выбрать соответственно 10^{-8} и 10^{-10}.

■ УПРАЖНЕНИЕ 2.38

В работе [Kamke, 1971, стр. 598] рассматривается ЗНУ

$$y(y'')^2 = e^{2x},\ y(0) = 0, y'(0) = 0,$$

с использованием которой можно описать пространственное распределение заряда в цилиндрическом конденсаторе. Решение аппроксимируется рядом

$$y(x) = x^p(a + bx + \ldots).$$

Определите p и a. Покажите, что эта ЗНУ имеет только одно решение указанного вида. Поскольку решение сингулярно в начале координат, численное интегрирование ОДУ должно быть дополнено результатами аналитической аппроксимации при малых x. Получите соответствующие результаты при

$$b = \frac{27}{40}a^{-2}.$$

[1] Прим. перев. — Самая яркая звезда (α) в созвездии Возничего, являющаяся двойной звездой.

Если ранее вы не сделали упражнение 1.7, в котором вам предлагалось представить это ОДУ в виде системы уравнений первого порядка, сделайте это сейчас. После этого, найдите численное решение этой системы с учетом того, что y' возрастает вблизи $x = 0$. Определите аппроксимации $y(x_0)$ и $y'(x_0)$ при $x_0 = 0.001$, вычислив значения ряда и его производной. Выполните интегрирование системы уравнений первого порядка на интервале $[x_0, 0.1]$ с принятыми по умолчанию значениями допустимых ошибок. Выведите график численного решения, полученного при интегрировании, а также график приближенного решения, полученного вычислением значения ряда.

Глава 3

ЗАДАЧИ С ГРАНИЧНЫМИ УСЛОВИЯМИ

§ 3.1. ВВЕДЕНИЕ

Решение системы обыкновенных дифференциальных уравнений (ОДУ) можно определить, задав значения всех его компонент в одной начальной точке при значении независимой переменной $x = a$. Эта точка, а также направление интегрирования определяют задачу с начальными условиями (ЗНУ). Во многих прикладных задачах решение определяется более сложным образом. В задаче с граничными условиями (ЗГУ) значения компонент решения или определенные соотношения между компонентами решения задаются в нескольких точках при соответствующих значениях независимой переменной, принадлежащих к определенному интервалу. При определенных условиях ЗНУ имеет единственное решение, но в случае ЗГУ вопрос о существовании и единственности решения более сложен. Подобно системе линейных алгебраических уравнений, ЗГУ может не иметь ни одного решения, может иметь единственное решение или может иметь несколько решений. Вследствие возможности существования множества решений, в численных процедурах решения ЗГУ используются задаваемые пользователем оценки решения (предположения о виде решения), которые позволяют конкретизировать выбор искомого решения. Часто для того, чтобы рассматриваемая ЗГУ имела решение, необходимо определить множество параметров. В этом случае каждому такому решению может соответствовать определенное множество параметров, конечное число возможных множеств параметров или бесконечное число возможных множеств параметров. При вызове численной процедуры решения ЗГУ пользователю необходимо задать оценки для этих множеств. В главе 1 были рассмотрены несколько примеров ЗГУ. В этой главе будут представлены дополнительные примеры, которые позволят читателю более глубоко понять существо этого вопроса.

Далее мы будем рассматривать двухточечные ЗГУ, которые могут быть представлены в виде системы ОДУ

$$y' = f(x, y, p) \tag{3.1}$$

с множеством граничных условий, определяемым равенством

$$0 = g(y(a), y(b), p), \tag{3.2}$$

где p — вектор неизвестных параметров. Эти параметры могут иметь определенный физический смысл, если уравнение (3.1) представляет собой модель какого-либо

физического явления, или же введение этих параметров может быть обусловлено особенностями выполнения процесса численного решения рассматриваемой ЗГУ. На практике коэффициенты ОДУ нередко могут быть сингулярными и задача может быть сформулирована на бесконечном интервале. ЗГУ с неизвестными параметрами могут быть непосредственно решены с использованием численной процедуры MATLAB `bvp4c`. Однако, в общем случае, необходимо переформулировать ЗГУ так, чтобы в ней не использовались неизвестные параметры. Для упрощения записи мы часто будем опускать p в уравнении (3.1) и равенствах (3.2).

В предыдущей главе, посвященной исследованию ЗНУ, мы показали, как можно представить ОДУ в виде системы уравнений первого порядка вида (3.1). Мы также отметили, что некоторые численные процедуры позволяют решать уравнения, представленные в виде $y'' = f(x, y)$. Уравнения подобного вида часто используются в некоторых научных областях и существуют специальные методы их численного решения. Аналогично, существуют численные процедуры решения ЗГУ, которые позволяют исследовать уравнения, порядок которых превышает 1. Эти процедуры обладают определенными преимуществами, но имеют более сложный интерфейс с пользователем. Численная процедура `bvp4c` имеет достаточно простой интерфейс, но допускает решение лишь систем первого порядка. Более того, все реализованные в среде программирования MATLAB численные процедуры решения ЗНУ требуют представления соответствующего ОДУ в форме системы первого порядка.

В случае, когда в формулировке ЗНУ неизвестные параметры отсутствуют, граничные условия называются *разделенными*, если каждое из равенств (3.2) задано лишь для одной граничной точки. Если одно из граничных условий связывает значения решения в обоих граничных точках, то эти граничные условия называются *неразделенными*. Численная процедура `bvp4c` позволяет решать задачи с неразделенными граничными условиями. Однако большинство других процедур не позволяют решить подобную задачу и поэтому исходную ЗГУ необходимо переформулировать в терминах разделенных граничных условий. В многоточечных ЗГУ граничные условия заданы для более чем двух точек. Специальные численные процедуры допускают решение многоточечных ЗГУ, но при использовании большинства других процедур (включая `bvp4c`) необходимо представить исходную ЗГУ в форме двухточечной ЗГУ.

Выбор граничных условий не всегда бывает очевиден. Далее мы обсудим этот вопрос более подробно, уделив особое внимание случаю, когда интервал интегрирования бесконечен. При решении ЗГУ, имеющих сингулярности в граничных точках или заданных на бесконечном интервале, всегда присутствует элемент искусства. Решение подобных проблем будет продемонстрировано на простых примерах, с использованием которых мы познакомим читателя с имеющимися в распоряжении исследователя средствами численного анализа ЗГУ. Более сложные задачи мы рассмотрим более детально в целях иллюстрации использования различных подходов.

Вообще говоря, решение ЗГУ — это более сложная задача, чем решение ЗНУ, и любая численная процедура может оказаться бесполезной, даже если пользователем были заданы адекватные предположения о виде решения и вполне достоверные оценки неизвестных параметров. Более того, численная процедура решения ЗГУ может успешно получить численное решение в случае, когда это решение не

должно существовать! В параграфе 3.4 мы кратко рассмотрим численные методы, используемые в численных процедурах решения ЗГУ. Несмотря на то, что численная процедура `bvp4c` доказала свою эффективность, ее нельзя считать универсальной и пригодной для решения всех задач. То же можно сказать и в отношении любой другой численной процедуры решения ЗГУ. В частности, вследствие того, что в `bvp4c` используется численный метод небольшого порядка, она не может быть применена при решении задач, требующих высокой точности вычислений, или в задачах, решения которых допускают резкие изменения значений.

После обсуждения численных методов решения ЗГУ мы рассмотрим примеры, в которых применяется численная процедура `bvp4c`. Эти примеры иллюстрируют попутно разрабатываемые теоретические результаты. Например, в них показывается, как можно переформулировать задачу так, чтобы ее можно было решить с использованием численных процедур. В примерах показано также, как можно разрешить проблемы, связанные с наличием сингулярностей на концах интервала интегрирования, а также в случаях, когда этот интервал бесконечен. Упражнения предназначены для тех же целей и читатель должен ознакомиться с их содержанием, даже если он не намерен решать их.

§ 3.2. ЗАДАЧИ С ГРАНИЧНЫМИ УСЛОВИЯМИ

Примеры главы 1 продемонстрировали, что методики решения ЗГУ и ЗНУ существенно отличаются. В этом параграфе мы рассмотрим другие обосновывающие это утверждение примеры, которые иллюстрируют подход, позволяющий выполнить анализ ЗГУ, путем проведения анализа соответствующей ЗНУ.

Если функция $f(x, y, y')$ является достаточно гладкой, то ЗНУ, определяемая уравнением

$$y'' = f(x, y, y')$$

и начальными условиями

$$y(a) = A,\ y'(a) = s,$$

имеет единственное решение $y(x)$ при $x \geqslant a$. В качестве примера двухточечной ЗГУ можно рассмотреть линейное уравнение

$$y'' + y = 0 \tag{3.3}$$

с разделенными граничными условиями

$$y(a) = A,\ y(b) = B.$$

При исследовании подобных задач $y(x, s)$ рассматривается в качестве решения уравнения (3.3) с начальными значениями $y(a, s) = A$ и $y'(a, s) = s$. Для каждого значения параметра s решение $y(x, s)$ удовлетворяет граничному условию в конечной точке $x = a$ и продолжимо от $x = a$ до $x = b$. Возникает вопрос: при каких значениях s решение $y(x, s)$ удовлетворяет другому граничному условию при $x = b$? Иными словами, при каких значениях s выполнено $y(b, s) = B$? Если для этого алгебраического уравнения существует решение s, то $y(x, s)$ представляет

собой решение рассматриваемого ОДУ, удовлетворяющее обоим граничным условиям, т. е. оно является решением исходной двухточечной ЗГУ. С учетом линейности уравнения, мы можем легко классифицировать возможные ситуации. Пусть $u(x)$ — решение уравнения (3.3), определенное начальными условиями $y(a) = A$ и $y'(a) = 0$ и пусть $v(x)$ — решение, определенное начальными условиями $y(a) = 0$ и $y'(a) = 1$. Общее решение уравнения (3.3) имеет следующий вид

$$y(x, s) = u(x) + sv(x).$$

С учетом граничного условия

$$B = y(b, s) = u(b) + sv(b)$$

получаем линейное алгебраическое уравнение, относительно неизвестного начального s. Из известных результатов о существовании и единственности решения линейного алгебраического уравнения следует, что при $v(b) \neq 0$ существует только одно решение ЗГУ и оно соответствует значению s

$$s = \frac{B - u(b)}{v(b)}.$$

Кроме того, если $v(b) = 0$, то при $B = u(b)$ существует бесконечно много решений, а при $B \neq u(b)$ — ни одного. Вопрос о существовании и единственности решений для этой ЗГУ достаточно очевиден, но при $v(b) \approx 0$ могут возникнуть определенные трудности вычислительного характера. Например, существует опасность вычисления решения в случае, когда не существует ни одного решения или численная процедура может придти к ошибочному заключению, что решения не существует.

Решение задач о собственных значениях, и, в частности, задачи Штурма–Лиувилля, мы проиллюстрируем на примере ОДУ

$$y'' + \lambda y = 0, \tag{3.4}$$

решение которого определяется либо при периодических граничных условиях

$$y(0) = y(\pi),\ y'(0) = y'(\pi),$$

являющихся неразделенными, либо при граничных условиях Дирихле

$$y(0) = 0,\ y(\pi) = 0,$$

являющихся разделенными. При всех значениях параметра λ функция $y(x) \equiv 0$ является решением этой ЗГУ, но при некоторых значениях λ существуют нетривиальные решения. Эти значения называются *собственными значениями*, а соответствующие нетривиальные решения — *собственными функциями*. Задача Штурма–Лиувилля заключается в вычислении собственных значений и собственных функций. Эта ЗНУ является нелинейной, т. к. неизвестный параметр λ умножается на неизвестное решение $y(x)$. Если $y(x)$ представляет собой решение рассматриваемой ЗГУ, то, как легко заметить, $\alpha y(x)$ также является решением этой ЗГУ при

любой константе α. Таким образом, необходимо задать нормировочное условие, определяющее то решение, которое нас интересует. Например, можно потребовать, чтобы было выполнено $y'(0) = 1$. Это условие можно считать всегда выполненным, поскольку условию $y'(0) = 0$ соответствует тривиальное решение, и, если $y'(0) \neq 0$, мы можем масштабировать $y(x)$ так, чтобы было выполнено $y'(0) = 1$. Нормировочное условие можно рассматривать как новое граничное условие, необходимое для определения неизвестного параметра λ и вводимое в дополнение к граничным условиям, определяющим решение $y(x)$ уравнения второго порядка (3.4).

При $\lambda > 0$ решение ЗНУ, удовлетворяющее уравнению (3.4) и начальным значениям $y(0) = 0$ и $y'(0) = 1$, имеет следующий вид

$$y(x) = \frac{\sin(x\sqrt{\lambda})}{\sqrt{\lambda}}.$$

В случае исследования рассматриваемой ЗГУ с граничными условиями Дирихле, начальное условие $y(\pi) = 0$ приводит к нелинейному алгебраическому уравнению относительно неизвестного параметра λ. Вопрос о существовании и единственности решений нелинейных алгебраических уравнений в общем случае весьма сложный. Однако в рассматриваемом случае это алгебраическое уравнение может быть легко решено. В результате мы приходим к заключению, что ЗГУ имеет нетривиальное решение, если и только если собственное значение определяется одним из следующих равенств

$$\lambda = k^2, \ k = 1, 2, \ldots$$

Этот пример показывает, что при решении задачи Штурма–Лиувилля следует указывать условия, которые определяют интересующие нас собственные значения. Задачи Штурма–Лиувилля имеют важное значение в приложениях и поэтому теория их решения хорошо разработана. В монографии [Pryce, 1993] детально обсуждается вопрос о том, как эта теория может быть применена при численных расчетах. Существуют достаточно эффективные численные процедуры, которые специально разработаны для решения задачи Штурма–Лиувилля. Среди них мы хотели бы отметить D02KDF [NAG, 2002], SL02F [Marletta & Pryce, 1995], SLEIGN [Bailey, Gordon & Shampine, 1978] и SLEDGE [Pruess, Fulton & Xie, 1992]. Эти численные процедуры обладают весьма развитым интерфейсом с пользователем. Например, вы можете потребовать, чтобы численная процедура вычислила пятое собственное значение (если оно является возрастающим) и соответствующую собственную функцию. Вы можете решить задачу Штурма–Лиувилля с использованием рассматриваемых в этой главе численных процедур, предназначенных для решения ЗГУ общего вида, но в этом случае необходимо задать критерии отбора интересующих вас собственного значения и собственной функции. При этом вам может помочь следующий теоретический результат: k-я собственная функция осциллирует k раз на интервале между граничными точками a и b. Однако в любом случае вы не можете быть уверены в том, что численная процедура действительно вычислит желаемое собственное значение и соответствующую собственную функцию.

Наличие нелинейности вносит в задачу дополнительные сложности. Рассмотрим, например, следующее ОДУ [Bailey, Shampine & Waltman, 1968]

$$y'' + |y| = 0$$

с разделенными граничными условиями

$$y(0) = 0,\ y(b) = B.$$

Если линейная ЗГУ имеет более одного решения, она имеет бесконечно много решений, но нелинейная ЗГУ в подобной ситуации может иметь конечное число решений. Выполняя анализ так же, как это было сделано в линейном случае, можно придти к заключению, что для любой точки $b > \pi$ и при каждом значении $B < 0$ существует точно два решения рассматриваемой ЗГУ. Первое решение имеет вид $y(x, s) = s\sinh(x)$. Оно начинается в начале координат, характеризуется отрицательным значением s и монотонно убывает до значения B в граничной точке $x = b$. Второе решение имеет вид $y(x, s) = s\sin(x)$, начинается в начале координат и характеризуется положительным значением s. Это решение пересекает ось абсцисс в точке $x = \pi$, в которой происходит изменение вида решения: после прохождения этой точки оно монотонно убывает до значения B в граничной точке $x = b$. На рисунке 3.1 показаны результаты численных расчетов при $b = 4$ и $B = -2$. При решении нелинейных ЗНУ еще более важно, чем при решении задач на собственные значения, задавать условия отбора интересующего вас решения. В демонстрационной программе MATLAB `twobvp`, с использованием которой был получен рисунок 3.1, ЗГУ решается дважды при различных критериях отбора решения: при вычислении отрицательного на промежутке $(0, 4\pi]$ решения в качестве этого критерия выступало условие $y(x) \equiv -1$, а при вычислении второго решения использовалось условие $y(x) \equiv 1$.

Представленные выше примеры являются предельно простыми, что обусловлено их предназначением — служить в качестве иллюстрации общих концепций. В последующих примерах мы рассмотрим более сложные модели реальных физических явлений, которым соответствуют ЗГУ с неединственными решениями. Кроме того, существуют примеры, которые показывают, что задачи, включающие физические параметры, могут иметь решения лишь в случаях, если значения этих параметров удовлетворяют некоторым условиям. Таким образом, на практике решение ЗГУ тесно связано с исследованием вопроса о существовании и единственности решения рассматриваемой модели. В этом смысле ЗГУ существенным образом отличается от ЗНУ, поскольку при анализе последней существование и единственность решения гарантируется нежесткими условиями, которые почти всегда оказываются выполненными на практике.

§ 3.3. ГРАНИЧНЫЕ УСЛОВИЯ

В ситуациях, когда модели какого-либо физического явления соответствует некоторая ЗГУ, не всегда бывает очевидным, какого рода граничные условия следует использовать и в каких точках их следует применить. В этом параграфе мы кратко обсудим этот вопрос и обратимся к связанной с ним проблеме сингулярности ЗГУ, когда соответствующие ОДУ являются сингулярными в конечных точках

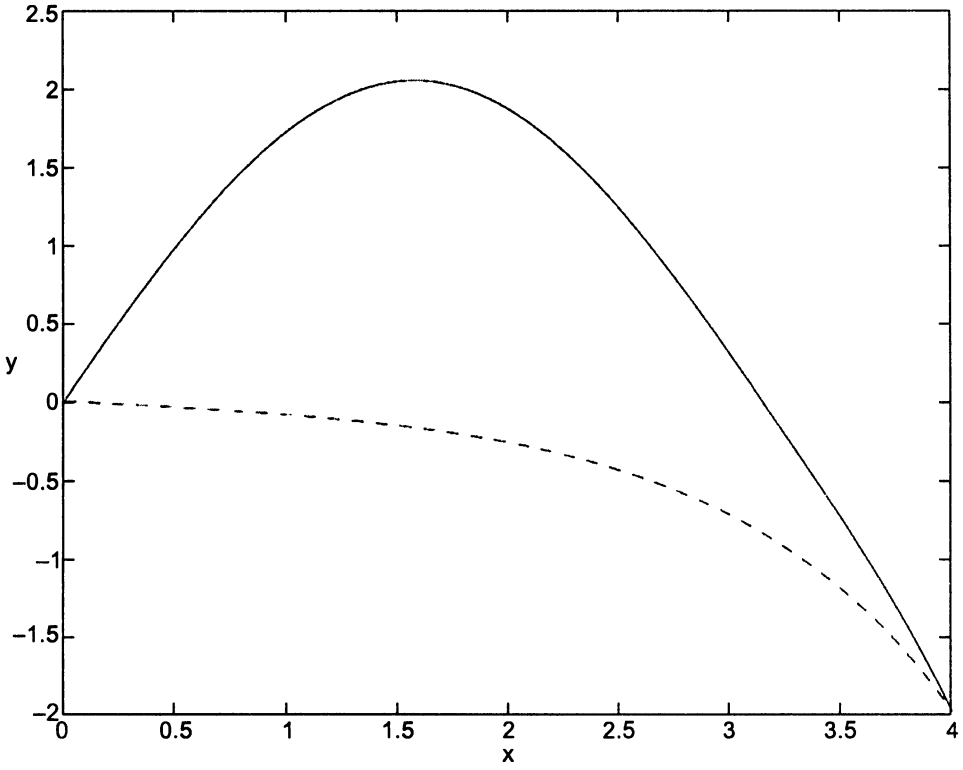

Рис. 3.1. Нелинейная ЗГУ, имеющая точно два решения.

или/и интервал интегрирования является бесконечным. Вообще говоря, для ЗГУ сингулярность является скорее типичной характеристикой, нежели особым случаем. Стандартные теоретические результаты оказываются неприменимыми в подобных ситуациях и поэтому при решении этих задач приходится использовать всю имеющуюся информацию о соответствующем физическом явлении и выполнять углубленный математический анализ рассматриваемой проблемы. В подпараграфах 3.3.1 и 3.3.2 мы рассмотрим несколько примеров, иллюстрирующих важные аспекты этой проблемы. В параграфе 3.5 детально рассматривается методика решения сингулярных задач путем их представления в форме, допускающей использование стандартных численных процедур. Другие способы разрешения проблемы сингулярности ЗГУ обсуждаются в упражнениях.

В общем случае, число граничных условий должно быть равно сумме порядков всех ОДУ, используемых в рассматриваемой ЗГУ, плюс число неизвестных параметров и в этом смысле ситуация полностью аналогична той, что возникает при решении ЗНУ. Для систем уравнений первого порядка, представляющих для нас наибольший интерес, число граничных условий должно быть равно числу ОДУ в системе плюс число неизвестных параметров. Численные процедуры решения ЗГУ не смогут выполнить требуемые вычисления, если вы зададите неверное число граничных условий (больше или меньше, чем требуется). Обычно граничные

условия определяются природой рассматриваемой задачи и в подобных ситуациях бывает достаточно просто определить их тип и точки, в которых они должны быть наложены. Однако нередки случаи, когда определение граничных условий связано с определенными трудностями. Это может произойти, например, если:

- Все физически обусловленные ограничения уже были использованы для определения граничных условий, но число последних по-прежнему меньше необходимого. В подобной ситуации исследователю следует задать дополнительные граничные условия, определяемые условиями сохранения некоторых интегралов движения.
- Некоторые граничные условия могут иметь специальный вид. Например, решение должно быть ограниченным в точке сингулярности, или при стремлении независимой переменной к бесконечности решение должно затухать, или затухать в соответствии с заданной характеристикой. В подобных ситуациях соответствующие требования должны и обычно могут быть преобразованы к виду стандартных граничных условий, например, с введением дополнительных неизвестных параметров. В этом параграфе, а также в параграфе 3.5, будут представлены примеры, иллюстрирующие эти два случая.
- Слишком много граничных условий может быть задано в случаях, когда ваше знание о рассматриваемой физической задаче настолько детально, что вы можете представить больше граничных условий, чем требуется для корректного определения ЗГУ. В подобных ситуациях следует отбросить те граничные условия, которые являются следствиями других граничных условий и специфики рассматриваемой системы ОДУ. Обычно (но не всегда) подобные граничные условия включают производные решения высшего порядка.

3.3.1. Граничные условия в сингулярных точках

В этом параграфе мы обсудим ситуацию, когда сингулярность имеет место в конечных точках. Проблемы подобного рода часто возникают при преобразовании ДУЧП в соответствующую систему ОДУ с учетом цилиндрической или сферической симметрии рассматриваемой модели. Например, уравнение Брату (Bratu)

$$\Delta y + e^y = 0$$

используется в качестве модели процесса воспламенения. В примере 3.5.1 исследуется несингулярный случай симметрии слоя, но с физической точки зрения цилиндрическая и и сферическая симметрии представляют еще больший интерес. В этих случаях в качестве интервала интегрирования выступает $[0, 1]$ и ОДУ принимает вид

$$y'' + k\frac{y'}{x} + e^y = 0, \tag{3.5}$$

где $k = 1$ или $k = 2$ в случае использования соответственно цилиндрической и сферической симметрии. Очевидно, что трудности, возникающие при анализе поведения (3.5) вблизи $x = 0$, связаны с наличием в уравнении члена y'/x. Разумно полагать, что при решении этой ЗГУ не должно возникать каких-либо проблем и полученное решение должно быть достаточно гладким, т. к. указанная сингулярность обусловлена не физическими причинами, а примененным преобразованием координат. Действительно, вследствие симметрии мы должны иметь $y'(0) = 0$ и,

следовательно, проблемный член скорее недоопределен, чем принимает бесконечное значение в указанной точке. В отношении этой и других подобных ЗГУ может быть применена стандартная теория сходимости численных методов [de Hoog & Weiss, 1976, 1978]. В случае использования методов, в которых вычисление ОДУ в $x = 0$ не выполняется, решение задач, подобных (3.5), может быть выполнено непосредственно. Очевидно, что методы, в которых выполняется вычисление этого уравнения в точке $x = 0$, использовать непосредственно невозможно. (В число этих методов ранее входила и процедура `bvp4c`, но начиная с версии MATLAB 6.5 эта процедура также может быть непосредственно использована при решении подобных задач.) Однако, рассматривая этот пример, мы стремились не столько обосновать тот факт, что некоторые численные процедуры не могут быть применены для решения подобных задач, сколько предложить методику, позволяющую решить проблему сингулярности в конечных точках.

Конструктивным решением этой проблемы является использование аналитической аппроксимации решения вблизи сингулярной точки. Реализация этой идеи в значительной степени зависит от специфики рассматриваемой задачи, а также от свойств решения, которые могут быть известны из физических соображений. Часто используются разложения в ряд или асимптотические разложения. Более подробно эти вопросы изложены в работе [Bender & Orszag, 1999], в которой также рассмотрено множество интересных примеров. При решении ЗНУ сингулярности рассматриваемого типа возникают сравнительно редко, но соответствующие проблемы решаются в целом аналогично и поэтому примеры и упражнения из подпараграфа 2.3.4 могут служить в качестве дополнительных примеров анализа сингулярных ситуаций.

В рассматриваемом здесь примере можно ожидать, что искомое решение является гладкой функцией и поэтому можно использовать разложение в ряд Тейлора. С учетом симметрии это разложение может быть записано следующим образом

$$y(x) = y(0) + \frac{y''(0)}{2}x^2 + \frac{y^{(4)}(0)}{4!}x^4 + \ldots$$

Подставив это выражение для $y(x)$ в ОДУ и используя граничные условия в $x = 0$, можно определить коэффициенты в этом разложении так, чтобы при $x \to 0$ соответствующее решение удовлетворяло ОДУ с максимально возможной точностью. Например,

$$(y''(0) + \ldots) + \frac{k}{X}(y''(0)x + \ldots) + e^{y(0)+} = 0.$$

Приравнивая нулю старшие (постоянные) члены, получаем необходимое для вычисления второго коэффициента ряда Тейлора значение второй производной

$$y''(0) = -\frac{e^{y(0)}}{k+1}.$$

Если опустить в ряде Тейлора члены более высокого порядка, полученная аппроксимация решения $y(x)$ достаточно точна на интервале $[0, \delta]$ при малых $\delta > 0$. Вычисляя производную этой аппроксимации, получаем приближенное значение для $y'(x)$. Далее находим численное решение ОДУ на интервале $[\delta, b]$, на котором

уравнение не является сингулярным. Каково граничное условие в $x = \delta$? Очевидно, что (аналитическое) приближенное решение на $[0, \delta]$ должно совпадать в точке $x = \delta$ с численным решением на $[\delta, b]$. Для рассматриваемого примера, должны быть выполнены равенства

$$y(\delta) = p - \frac{e^p}{2(k+1)}\delta^2,$$
$$y'(\delta) = -\frac{e^p}{k+1}\delta,$$

где $p = y(0)$ рассматривается как неизвестный параметр, который подлежит определению в процессе решения ЗГУ. Этот случай представляет собой пример, когда для решения проблемы сингулярности вводится дополнительный неизвестный параметр.

Граничное условие определяет то, как ведет себя интересующее нас решение при приближении к конечной точке. При этом мы подразумеваем не только достижение решением определенного значения. Для описания более сложного поведения решения мы будем использовать стандартные обозначения: выражение

$$f(x) \sim F(x)$$

при $x \to x_0$ означает, что

$$\lim_{x \to x_0} \frac{f(x)}{F(x)} = 1.$$

Это выражение читается следующим образом: «$f(x)$ является асимптотикой для $F(x)$ при x, стремящемся к x_0». Из граничного условия $y'(0) = 0$ следует, что решение задачи Брату стремится к константе при $x \to 0$. Однако это ОДУ может быть исследовано и при других граничных условиях. Рассмотрим, в частности, решения, которые не являются ограниченными в начале координат. Пусть $k = 1$. В предположении, что при $x \to 0$ производные решения растут быстрее самого решения, можно использовать следующую аппроксимацию исследуемого ОДУ

$$0 = y'' + \frac{y'}{x} + e^y \sim y'' + \frac{y'}{x}.$$

Решая это уравнение, получаем

$$y(x) \sim b + a\log(x) = b + \log(x^a)$$

где a и b — константы. Подставим эту аналитическую аппроксимацию в ОДУ для того, чтобы выяснить условия, при которых это асимптотическое решение действительно может удовлетворять исходному ОДУ

$$y'' + \frac{y'}{x} + e^y \sim -\frac{a}{x^2} + \frac{a}{x^2} + x^a e^b.$$

Таким образом, для того, чтобы решение указанного вида асимптотически удовлетворяло ОДУ, необходимо $a > 0$. Граничные условия вида $y(x) \sim \log(x)$ при $x \to 0$ возникают, например, при исследовании потенциальных проблем с зажиганием,

реализованным по линейной схеме. Итак, мы убедились в том, что рассматриваемое ОДУ может иметь решение с описанным поведением и поэтому указанное граничное условие допустимо. Аналогично, при $k = 2$ имеем

$$y(x) \sim b + ax^{-1}$$

и для того, чтобы в асимптотике это решение удовлетворяло ОДУ, необходимо $a < 0$. Граничные условия вида $y(x) \sim -x^{-1}$ возникают, например, при исследовании потенциальных проблем с зажиганием, реализованным по точечной схеме. Решение уравнения Брату с граничным условием $y(x) \sim \log(x)$ может быть выполнено аналогично решению этого уравнения с граничным условием для симметричного случая: на интервале $(0, \delta]$ при малой δ используются аналитические аппроксимации $y(x) \approx \log(x)$ и $y'(x) \approx x^{-1}$, а на интервале $[\delta, 1]$ ОДУ численно решается с граничным условием $y(\delta) = \log(\delta)$. Заметим, что при выборе этого граничного условия, дополнительный неизвестный параметр $y(0)$ не вводится, как это делалось в случае граничного условия для симметричного случая.

Задача определения поведения решений в сингулярной точке может оказаться более сложной. В работе [Bender & Orszag, 1999, стр. 170-2] рассматривается ОДУ

$$yy'' = -1 \tag{3.6}$$

с граничными условиями $y(0) = 0$ и $y(1) = 0$. Если интересующее нас решение должно удовлетворять требованию $y(x) \to 0$ при x, стремящемся к 0 и 1, то из ОДУ следует, что значение $y''(x)$ должно быть неограниченным в этих точках. Очевидно, что в этом случае при вычислении требуемого решения численная процедура должна использовать не только предельные значения для $y(x)$, но и некоторую дополнительную информацию, которую должен предоставить пользователь. В указанной работе была предложена асимптотическая аппроксимация для $y(x)$ и ниже мы используем этот результат. Поскольку решение симметрично относительно $x = 0.5$, нам необходимо исследовать его поведение лишь вблизи начала координат. Можно ожидать существование (по крайней мере) двух решений рассматриваемой задачи, поскольку если $y(x)$ — ее решение, то $y(x)$ также является ее решением. Поскольку решение обращается в ноль в начале координат и имеет при этом бесконечное значение производной, будем искать его в виде

$$y(x) \sim ax^b,$$

где a и b — константы. Подставляя это выражение в ОДУ, получаем

$$-1 = y(x)y''(x) \sim (ax^b)(b(b-1)x^{b-2}) = ab(b-1)x^{2b-2}.$$

Если $2b - 2 < 0$, то правая часть этого равенства не имеет предела при $x \to 0$. Если $2b - 2 > 0$, предел существует, но равен нулю. Если $2b - 2 = 0$, правая часть тождественно равна нулю. Таким образом, мы убедились, что решение указанного вида не может быть решением рассматриваемого ОДУ даже асимптотически. Попытаемся решить задачу, выбрав в качестве решения следующее

$$y(x) \sim ax(-\log(x))^b.$$

Подставляя это выражение в ОДУ, получаем

$$-1 = y(x)y''(x) \sim -a^2 b(-\log(x))^{2b-1}[1-(b-1)(-\log(x))^{-1}].$$

Если мы хотим, чтобы правая часть имела предел при $x \to 0$, необходимо $2b - 1 = 0$, т. е. $b = 0.5$. Тогда в этом пределе $a = \pm\sqrt{2}$ и существует точно два решения. Первое из них является положительным при положительных x и имеет вид $y(x) \sim x\sqrt{-2\log(x)}$. Для выполнения численного интегрирования исходного уравнения второго порядка оно представляется в виде системы уравнений первого порядка с введением новой переменной для $y'(x)$. Поскольку она принимает бесконечное значение в начале координат, необходимо учитывать особенности ее поведения вблизи этой точки. Однако, все положительные решения рассматриваемого ОДУ, обращающиеся в начале координат в ноль, ведут себя одинаково, если ограничиться рассмотрением лишь первых нескольких членов их разложения и поэтому нет необходимости во введении нового параметра, если требуется умеренная точность вычислений. В упражнении 3.4 читателю предлагается убедиться в том, что с учетом всей этой информации о поведении решения вблизи сингулярной точки, численное решение рассматриваемой ЗГУ может быть легко получено.

Для понимания физической природы систем, рассмотренных в последующих примерах этой главы, вам потребуется выполнить нижеследующие упражнения, в которых рассматриваются различные методики анализа поведения решений. В этой связи особенно важно выполнить упражнение 3.2. В примере 3.5.4 и упражнении 3.9 рассматриваются сингулярные граничные условия в начале координат и эти задачи усложнены тем, что их решение ищется на бесконечном интервале. После ознакомления с практикой использования предназначенных для решения ЗГУ численных процедур MATLAB вы можете вернуться к упомянутым упражнениям и выполнить численное интегрирование соответствующих ОДУ с граничными условиями.

■ УПРАЖНЕНИЕ 3.1

Если решение ЗГУ является гладким в сингулярной точке, можно отказаться от использования его аналитической аппроксимации в окрестности этой точки. Для иллюстрации этого утверждения рассмотрим задачу Штурма–Лиувилля для уравнения Лацко (Latzko's equation)

$$\frac{d}{dt}\left((1-x^7)\frac{dy}{dx}\right) + \lambda x^7 y = 0$$

с граничными условиями $y(0) = 0$ и конечным $y(1)$. Эта задача имеет физический смысл и интересна наличием сингулярности. Существует множество результатов, касающихся численного решения этого уравнения. В частности, в работе [Scott, 1973, стр. 153] приведены аппроксимации первых трех собственных значений и выполнено сравнение аналитических значений с численными 8.728, 152.45 и 435.2. Для того, чтобы записать это уравнение в виде системы уравнений первого порядка, удобно ввести переменные $y(x)$ и $v(x) = (1-x^7)y'(x)$. Тогда

$$y' = \frac{v}{1-x^7},$$
$$v' = -\lambda x^7 y.$$

Как и в любой другой задаче на нахождение собственных значений, нам необходимо задать для решения нормировочное условие (например, $y(1) = 1$). Дифференциальное уравнение сингулярно в $x = 1$. Если интересующее нас решение характеризуется тем, что значение $y'(1)$ конечно, то из первого уравнения вышеприведенной системы следует, что в этом случае должно быть выполнено $v(1) = 0$. Используя второе уравнение системы и правило Лопиталя, получаем

$$y'(1) = \lim_{x\to 1} \frac{v(x)}{1-x^7} = \lim_{x\to 1} \frac{v'(x)}{-7x^6} = \lim_{x\to 1} \frac{-\lambda x^7 y(x)}{-7x^6} = \frac{\lambda}{7}.$$

Мы можем численно решить рассматриваемую систему с граничными условиями $y(0) = 0$, $y(1) = 1$, $v(1) = 0$ при условии, что можно успешно вычислить $y(x)$ и $v(x)$ в $x = 1$. При решении ЗГУ с неизвестным параметром численная процедура `bvp4c` вызывает функцию вычисления ОДУ в точке сетки x и при соответствующих значениях аппроксимаций для $y(x)$ и λ. Если $x = 1$, эта функция должна возвращать предельное значение для $y'(1)$. Программный код этой функции должен быть написан с учетом того обстоятельства, что численная процедура всегда вычисляет ОДУ в конечных точках интервала. Если решение является гладкой функцией, при выполнении вычислений не потребуется вычислять ОДУ в точках x, расположенных настолько близко к сингулярной точке, что возникают трудности при вычислении $y'(x)$, т. е. те трудности, которые возникали в предыдущих примерах и преодолевались с использованием аналитических аппроксимаций вблизи сингулярности. Разумеется, критерий для переменной $v(x)$, задаваемый при вызове численной процедуры, должен быть согласован с поведением решения вблизи $x = 1$. Это требование может быть легко обеспечено, путем задания критерия для переменной $y(x)$ так, чтобы величина $(1-x^7)y'(x)$ выступала в качестве критерия для $v(x)$.

Другой метод выполнения анализа в рассматриваемом примере основан на поиске решения в виде $y(x) \sim 1 + c(x-1)$ при $x \to 1$. Если решение подобного вида существует, то $y'(x) \sim c$. Поскольку для решения указанного вида величина $y'(1)$ конечна, выполнено $v(1) = 0$. Определите константу c путем подстановки этого решения в ОДУ. Покажите, что $(1-x^7) \sim 7(1-x)$ вблизи $x = 1$. При надлежащей программной реализации процедуры вычисления ОДУ рассматриваемая задача может быть легко решена с использованием численной процедуры `bvp4c`. Однако задача вычисления *определенных* собственных значений и собственных функций более сложна. Это обусловлено тем, что не всегда понятно, какое из решений будет получено при указанных критериях для $y(x)$ и λ и вообще возможно ли получение решения при таких условиях. Ранее упоминались специальные численные процедуры решения задачи Штурма–Лиувилля, основанные на более глубоких теоретических результатах в этой области. Читателю предлагается с использованием `bvp4c` попытаться получить одну или несколько аппроксимаций собственных значений, вычисленных в вышеупомянутой работе [Scott, 1973]. При выполнении соответствующих расчетов мы обнаружили, что в условиях предположения о виде решения $y(x) \approx x\sin(2.5\pi x)$ и при задании начальной сетки с использованием функции `linspace(0,1,10)` при $\lambda = 10$, 100, 500, 1000, 5000 и 10 000 имеет место сходимость к различным собственным значениям, причем в их число входят три значения, вычисленные в работе [Scott, 1973]. Этот результат представляется

удивительным, поскольку при других предположениях о виде решения $y(x)$ или даже при другом выборе начальной сетки, при указанных значениях оценок параметра λ мы не получали различных собственных значений.

■ УПРАЖНЕНИЕ 3.2

В документации к численной процедуре `D02HBF`, входящей в состав библиотеки программ NAG [NAG, 2002] и предназначенной для решения ЗГУ, рассматривается следующая задача

$$2xy'' + y' = y^3,\ y(0) = \frac{1}{10},\ y(16) = \frac{1}{6}.$$

Исследуйте решение, которое ведет себя при $x \to 0$ как

$$y(x) \sim \alpha + px^{\beta} + \gamma x,$$

где $0 < \beta < 1$. Определите α, β и γ из ОДУ и граничного условия в точке $x = 0$. Коэффициент p выступает в роли неизвестного параметра. Решение, поведение которого удовлетворяет описанным выше свойствам, имеет при $x \to 0$ неограниченную первую производную. При численном решении этой ЗГУ необходимо использовать асимптотические аппроксимации для $y(x)$ и $y'(x)$ на интервале $(0, d]$ при малых d, а на интервале $[d, 16]$ необходимо выполнить стандартное численное интегрирование. Равенство в точке d численного значения и значения асимптотического решения обеспечивает выполнение граничных условий для рассматриваемой ЗГУ. Для того, чтобы определить неизвестный параметр p, необходимо потребовать непрерывность функций $y(x)$ и $y'(x)$. Асимптотические аппроксимации этих функций, а также оценки для p, следует использовать в качестве критериев отбора решения на интервале $[d, 16]$.

■ УПРАЖНЕНИЕ 3.3

В работе [Keller, 1992, раздел 6.2] рассматривается численное моделирование изменения концентраций субстрата в реакции ферментативного катализа с кинетикой Михаэлиса–Ментен (Michaelis–Menten kinetics). Задача рассматривалась в сферической области, поэтому с учетом этой симметрии соответствующее ДУЧП может быть сведено к ОДУ

$$y'' + 2\frac{y'}{x} = \frac{y}{\varepsilon(y + k)}.$$

Это уравнение содержит два параметра: ε и k. Задача рассматривается при граничных условиях $y(1) = 1$ и $y'(0) = 0$ (условие симметрии). Используя численную процедуру `bvp4c`, численно решите эту задачу при значениях параметров $\varepsilon = 0.1$ и $k = 0.1$. Аппроксимируйте $y(x)$ на интервале $[0, d]$ рядом Тейлора, ограничившись несколькими членами, и выполните численное интегрирование на интервале $[d, 1]$. При выводе графика решения на всем интервале $[0, 1]$ дополните результаты численного интегрирования на $[d, 1]$ значениями, вычисленными в $x = 0$ при значении неизвестного параметра $p = y(0)$ и начальном значении производной $y'(0) = 0$. При $d = 0.001$ достаточно использовать два и один член в рядах Тейлора, аппроксимирующих соответственно функции $y(x)$ и $y'(x)$. При написании программы вам необходимо каким-либо образом передать в процедуру вычисления ОДУ граничные

условия, а также значение d и параметры ε и k. Это можно сделать либо путем введения соответствующих переменных в тексте этой процедуры, либо завести соответствующие глобальные переменные, либо передать эти значения как опциональные входные переменные численной процедуры `bvp4c`. В последнем случае следует иметь в виду, что эти опциональные аргументы должны быть перечислены в списке входных аргументов после неизвестного параметра p. Кроме того, если вы не задаете никаких опций процедуры `bvp4c`, в качестве этого аргумента следует использовать `[ ]` и после него должен следовать список опциональных аргументов (начинающийся, как было отмечено выше, с p).

■ УПРАЖНЕНИЕ 3.4

Используя численную процедуру `bvp4c`, численно решите ЗГУ

$$yy'' = -1,\ y(0) = 0,\ y'(0.5) = 0.$$

Можно показать, что $y(x) \sim x\sqrt{-2\log(x)}$ при $x \to 0$. Напишите программный код процедуры, предназначенной для вычисления аппроксимации $v(x) = x\sqrt{-2\log(x)}$. Переместите граничное условие из сингулярной точки в начале координат в $d = 0.001$, потребовав $y(d) = v(d)$. Вычислите $y(x)$ на $[d, 0.5]$, используя численную процедуру `bvp4c` с заданными по умолчанию значениями допустимых ошибок и критериями отбора $y(x) \approx x(1-x)$ и $y'(x) \approx 1 - 2x$. Выведите графики $y(x)$ и $v(x)$ на координатной плоскости, заданной функцией `axis([0 0.5 0 0.5])` (см. рис. 4.11 в работе [Bender & Orszag, 1999]). Для этого дополните полученные в результате численного интегрирования массивы $y(x)$ и $v(x)$ значениями $y(0) = 0$ и $v(0) = 0$ соответственно.

3.3.2. Граничные условия, заданные на бесконечности

Корректность постановки ЗГУ зависит от природы решения соответствующего ОДУ, а также от реализуемости заданных граничных условий. В некоторых случаях граничные условия могут быть установлены с учетом физики задачи, но если это не представляется возможным, следует быть готовым к возникновению различных проблем. В особенности это утверждение верно, если задача поставлена на бесконечном или даже на ограниченном, но достаточно *большом* интервале. После ознакомления с видами возможных граничных условий на бесконечности мы рассмотрим несколько примеров, в которых показывается, как можно переформулировать задачу так, чтобы для ее решения можно было бы использовать численные процедуры, предназначенные для задач, поставленных на конечном интервале. Применяемый при этом подход аналогичен тому, что используется в случаях сингулярности в конечных граничных точках.

Рассмотрим в качестве иллюстрирующего примера уравнение

$$y''' + 2y'' - y' - 2y = 0. \tag{3.7}$$

Его общее решение имеет следующий вид

$$y(x) = Ae^{x} + Be^{-x} + Ce^{-2x}.$$

Заметим, что решение состоит из трех компонент: первая возрастает, а две последние убывают при увеличении x. Предположим, что нам необходимо найти решение этого уравнения на интервале $[0, +\infty)$ с граничными условиями

$$y(0) = 1,\ y'(0) = 1,\ y(+\infty) = 0.$$

Из первого граничного условия

$$y(\infty) = 0 = A \times \infty + B \times 0 + C \times 0$$

следует, что $A = 0$. Из оставшихся двух граничных условий

$$y(0) = 1 = A \times 1 + B \times 1 + C \times 1,$$
$$y'(0) = 1 = A \times 1 - B \times 1 - 2C \times 1$$

следует, что $B = 3$ и $C = -2$. Таким образом, решение этой ЗГУ существует и единственно. С другой стороны, если в качестве граничных условий рассмотреть

$$y(0) = 1,\ y(+\infty) = 0,\ y'(+\infty) = 0, \tag{3.8}$$

то из второго граничного условия $y(\infty) = 0$ следует, что $A = 0$, но последнее граничное условие

$$y'(\infty) = 0 = A \times \infty - B \times 0 - 2C \times 0$$

не накладывает никаких ограничений на коэффициенты в выражении для решения. Из оставшегося неиспользованным граничного условия

$$y(0) = 1 = A \times 1 + B \times 1 + C \times 1$$

следует, что $C = 1 - B$. Таким образом, рассматриваемая задача с указанными граничными условиями имеет бесконечно много решений, каждое из которых соответствует определенному значению B.

Если задача с граничными условиями, заданными в бесконечной точке $b = +\infty$, оказывается некорректно поставленной, то можно ожидать, что при численном решении этой же задачи, но при граничных условиях, заданных в конечной точке $b \gg +1$, могут возникнуть проблемы. Предположим, что необходимо решить уравнение (3.7) с граничными условиями

$$y(0) = 1,\ y(b) = 0,\ y'(b) = 0.$$

При больших значениях b система линейных уравнений, определяющая значения коэффициентов A, B и C в выражении для общего решения, является плохо обусловленной. Действительно, даже при значении b, близком к 20, попытка численного обращения в MATLAB соответствующей матрицы приводит к выводу сообщения о том, что эта матрица близка к сингулярной; при этом выдается оценка ее числа обусловленности $5 \cdot 10^{17}$=`1/RCOND`. Существо возникающей проблемы заключается в том, что две экспоненциально быстро убывающие с ростом x компоненты решения ОДУ численно неразличимы в точке $x = b$ и, следовательно, линейная система,

используемая для нахождения коэффициентов линейной комбинации, определяющей искомое решение ЗНУ, численно сингулярна. Первая компонента решения e^x экспоненциально быстро убывает с уменьшением x, что должно привести к плохой обусловленности линейной системы в случаях, когда b велико и среди граничных условий имеется условие, соответствующее значению этой компоненты в точке $x = 0$. В упражнении 3.5 рассматривается ситуация, соответствующая этому случаю.

Рассмотрим более общую задачу, включающую рассмотренный выше пример. Система линейных ОДУ с постоянными коэффициентами имеет следующий вид

$$y' = Jy + q. \tag{3.9}$$

Предположим для простоты изложения, что матрица Якоби J невырождена. Тогда $p(x) = -J^{-1}q$ — постоянное частное решение рассматриваемой системы ОДУ. Предположим также, что J имеет полную систему собственных векторов $\{v^j\}$ с соответствующими собственными значениями $\{\lambda_j\}$. В этом случае существуют константы α_j, такие что

$$y(a) - p(a) = \sum \alpha_j v^j.$$

Можно показать, что общее решение рассматриваемой системы ОДУ имеет следующий вид

$$y(x) = p(x) + \sum \alpha_j e^{\lambda_j (x-a)} v^j.$$

Легко видеть, что решение $y(x)$ конечно на $[a, +\infty)$ только в том случае, если из граничных условий в точке $x = a$ следует $\alpha_m = 0$ для всех m, таких что $\mathrm{Re}(\lambda_m) > 0$. Соответственно, ЗГУ является корректно определенной при $b \gg a$, только если среди граничных условий в $x = a$ нет условия, соответствующего компоненте решения, экспоненциально возрастающей при увеличении x. Аналогично, используя разложение

$$y(b) - p(b) = \sum \beta_j v^j,$$

можно показать, что задача является корректно поставленной при $b \gg a$, если из граничных условий в $x = b$ следует $\beta_m = 0$ для любого значения m, такого что $\mathrm{Re}(\lambda_m) < 0$. Нестрого говоря, компоненты решения, убывающие с ростом x, должны определяться граничными условиями на левом конце интервала, а компоненты решения, возрастающие с ростом x, должны определяться граничными условиями на правом конце интервала.

Несмотря на то, что рассматриваемое здесь линейное уравнение с постоянными коэффициентами является частным случаем систем ОДУ общего вида, с использованием этого примера можно уяснить для себя важные аспекты решения ЗГУ. Например, мы выяснили, что граничные условия определяют в определенной степени возможное поведение решений. Если ОДУ имеет решения, которые быстро приближаются друг к другу с ростом x на промежутке от a до b, то при $b \gg a$ эти решения будут численно неразличимы на правом конце интервала $x = b$. Поэтому критерии отбора для подобных решений должны быть основаны на использовании граничных условий не в точке $x = b$, а в точке $x = a$. Имеет место *дихотомия* пространства решений ОДУ и это обстоятельство должно учитываться при выборе точек, в которых задаются граничные условия. Это может быть сравнительно

легко сделано в случае линейного ОДУ с постоянными коэффициентами, но в общем случае для адекватного выбора типа граничных условий, а также точек, в которых они задаются, необходимо использовать дополнительную информацию о физике рассматриваемой задачи. Более детально этот вопрос рассматривается в работе [Ascher, Mattheij & Russell, 1995], в которой приводятся также несколько иллюстративных примеров.

Решения бегущей волны для уравнения Фишера имеют важное практическое значение. Соответствующие теоретические результаты изложены в работе [Murray, 1993, раздел 11.2]. Волна, движущаяся со скоростью c, описывается функцией вида $u(x,t) = U(z)$, где $z = x - ct$ и функция $U(z)$ удовлетворяет ОДУ

$$U'' + cU' + U(1-U) = 0. \tag{3.10}$$

Очевидно, что это уравнение имеет два тривиальных решения (стационарных состояния): $U(z) \equiv 1$ и $U(z) \equiv 0$. В качестве граничных условий для решения волнового фронта обычно рассматриваются

$$U(-\infty) = 1,\ U(+\infty) = 0.$$

(Установившиеся состояния не удовлетворяют одновременно этим граничным условиям.) Таким образом, можно ожидать, что используя эти граничные условия и задав скорость движения волны c, мы должны придти к корректно сформулированной ЗГУ. Однако нетрудно заметить, что эта цель не может быть достигнута, поскольку рассматриваемое ОДУ имеет бесконечно много решений. Действительно, если $U(z)$ — решение, то для любой константы γ функция $U(z+\gamma)$ также является решением рассматриваемой задачи. При численном исследовании этой ЗГУ представляется заманчивым выбрать некоторое большое число Z, задать граничные условия $U(-Z) = 1$ и $U(Z) = 0$, заменив ими соответствующие граничные условия в точках $-\infty$ и $+\infty$, и использовать стандартную численную процедуру решения ЗГУ. Однако маловероятно, что этот подход приведет к успеху, поскольку ЗГУ имеет бесконечно много решений, но мы не предоставили критерия, с использованием которого можно было бы их различить. В указанной работе исследование этой ЗГУ было выполнено в фазовом пространстве (U, U'). Установившимся состояниям соответствуют сингулярные точки $(0,0)$ и $(1,0)$. Анализ устойчивости системы показал, что при $c \geqslant 2$ особые точки $(0,0)$ и $(1,0)$ являются соответственно устойчивым узлом и седловой точкой. Из анализа траекторий в фазовом пространстве следует, что при $c \geqslant 2$ существует решение ЗГУ, поведение которого характеризуется следующими свойствами: при приближении к точке $(0,0)$

$$U'(z) \sim \beta U(z),$$

где $\beta = (-c + \sqrt{c^2 - 4})/2$ и при приближении к точке $(1,0)$

$$(U(z) - 1)' \sim \alpha(U(z) - 1),$$

где $\alpha = (-c + \sqrt{c^2 + 4})/2$.

Анализ устойчивости в фазовом пространстве предполагает аппроксимацию рассматриваемого уравнения линейным дифференциальным уравнением с постоянными коэффициентами, имеющим вид (3.9). Однако исследование устойчивости

может быть выполнено непосредственно. В условиях предположений, исключающих вырожденные случаи, решение уравнения (3.9) представляет собой сумму постоянного частного решения и линейной комбинации экспоненциальных решений однородного уравнения. Найдем решение подобного вида, приближающееся к точке $(1,0)$ при $z \to -\infty$. В частности, найдем решение вида

$$U(z) \sim 1 + pe^{\alpha z}$$

и, соответственно,

$$U'(z) \sim \alpha p e^{\alpha z}.$$

Если точка $(U(z), U'(z))$ приближается к этому стационарному состоянию, должно быть выполнено $\mathrm{Re}(\alpha) > 0$. Подставляя выражение для решения в ОДУ, получаем

$$U'' + cU' = (U-1)U,$$
$$\alpha^2 p e^{\alpha z} + c\alpha p e^{\alpha z} \sim p e^{\alpha z} + p^2 e^{2\alpha z}.$$

Таким образом, необходимо найти такое значение α, при котором

$$\alpha^2 + c\alpha \sim 1 + pe^{\alpha z} \sim 1$$

при $z \to -\infty$. Из этого выражения получаем $\alpha = (-c \pm \sqrt{c^2+4})/2$. То, что мы получили два значения, отражает тот факт, что указанное устойчивое состояние является седловой точкой в фазовом пространстве. Положительное значение $\alpha = (-c + \sqrt{c^2+4})/2$ соответствует решению с требуемыми свойствами. Далее необходимо выбрать граничные условия, которые были бы совместимы с надлежащим поведением решения при $z \to -\infty$. Мы определили скорость убывания, но нам неизвестен множитель p, величина которого зависит от искомого решения. Поэтому мы должны рассматривать p либо как неизвестный параметр, либо переопределить условия так, чтобы они не зависели от p. Простейший способ обеспечить это требование заключается в наложении требования

$$\frac{U'(z)}{U(z)-1} \to \alpha.$$

Таким образом, можно выбрать некоторое число $Z \gg 1$ и рассмотреть

$$\frac{U'(-Z)}{U(-Z)-1} - \alpha = 0$$

в качестве граничного условия на левом конце интервала.

В иллюстративных целях можно рассмотреть несколько другой подход, позволяющий по иному определить поведение $U(z)$ при $z \to +\infty$. Поскольку $U(z)$ стремится к нулю, при больших z функцию $U(z)(1-U(z))$ можно аппроксимировать функцией $U(z)$. Соответствующая аппроксимация уравнения (3.10) имеет следующий вид

$$U'' + cU' + U = 0.$$

Решая это уравнение, получаем

$$U(z) \sim qe^{\beta z},$$

где β — корень уравнения $\beta^2 + c\beta + 1 = 0$. Оба корня этого квадратного уравнения отрицательны и неясно, какой из них следует выбрать. Более детальный анализ, выполненный в упомянутой выше работе, показал, что следует использовать $\beta = (-c + \sqrt{c^2 - 4})/2$. По существу мы можем ограничиться рассмотрением сравнительно медленно убывающей компоненты решения, т. к. в этом случае более быстро убывающая компонента решения гарантированно обратится в ноль в граничной точке. Как и ранее, существует проблема определения множителя q и формы граничного условия. Напомним, что если $U(z)$ является решением ОДУ, то $U(z + \gamma)$ также является решением этого уравнения. В качестве возможного варианта разрешения этой проблемы, можно выбрать конкретное значение q и вычислить частное решение. Как и в указанной работе, выберем $q = 1$ и наложим граничное условие на правом конце интервала

$$\frac{U(Z)}{e^{\beta Z}} - 1 = 0.$$

Продолжение решения этой задачи рассматривается в примере 3.5.5; см. также упражнения 3.25 и 3.26.

На основании представленных результатов у читателя может сложиться впечатление, что если ЗГУ поставлена на бесконечном интервале, ее решения всегда экспоненциально затухают. Однако это не так и существуют практические задачи, решения которых характеризуются другими свойствами. В качестве простого примера рассмотрим уравнение

$$y' = \lambda x y$$

с граничными условиями $y(0) = 1$ и $y(x) \sim e^{-x^2}$ при $x \to \infty$. Здесь λ — неизвестный параметр, который может быть определен аналитически путем решения ОДУ с начальным значением $y(0) = 1$ с последующим наложением граничного условия на бесконечности. В результате, можно показать, что решение имеет вид

$$\lambda = -2, \quad y(x) = e^{-x^2}.$$

При решении задач на бесконечном интервале граничные условия на бесконечности часто заменяются на граничные условия при большом значении b. Многие задачи, сформулированные на бесконечном интервале, имеют экспоненциально затухающие решения, и в этих случаях нередко в качестве значения b выбирают слишком большую величину, не принимая в расчет то обстоятельство, что решения рассматриваемой задачи затухают очень быстро. В результате этого, численная процедура решения ЗГУ может завершиться с ошибкой, т. к. она не сможет численно различить различные решения вблизи $x = b$.

С другой стороны, существуют ЗГУ, решения которых также затухают и имеют вид рациональных функций (т. е. эти решения затухают не экспоненциально). В качестве подобного простого примера можно рассмотреть уравнение

$$y' = -\lambda y^2$$

с граничными условиями $y(0) = 1$ и $y(+\infty) = 0$; константа λ выступает в роли неизвестного параметра. Вычисленное аналитически решение этого ОДУ,

удовлетворяющее граничному условию $y(0) = 1$, имеет следующий вид

$$y(x) = \frac{1}{\lambda x + 1}.$$

Это решение удовлетворяет граничному условию на бесконечности для любой $\lambda > 0$, т. е. рассматриваемая ЗГУ имеет бесконечно много решений. Несмотря на то, что этот пример не имеет физического смысла, он показывает, что для корректной постановки сингулярной задачи важно использовать информацию о поведении ее решений. В рассматриваемом в этом примере случае следует задать граничное условие $y(x) \sim x^{-1}$ при $x \to +\infty$ и показать, что при $\lambda = 1$ существует единственное решение. В условиях, когда решение характеризуется указанными свойствами затухания, в качестве бесконечного предела приходится использовать довольно большие значения b. Действительно, предположим, что мы заменили указанное граничное условие на $y(b) = 1/b$ при некотором большом b. Тогда можно показать, что решение на этом конечном интервале имеет вид

$$\lambda_b = 1 - \frac{1}{b}, \quad y_b(x) = \frac{1}{x + 1 - x/b}.$$

Легко видеть, что для того, чтобы решение на соответствующем интервале $[0, b]$ совпадало (даже с умеренной точностью) с решением задачи, поставленной на интервале $[0, \infty)$, необходимо использовать очень большое значение b.

Для рассмотрения в дальнейшем примеров, связанных с конкретными физическими процессами, полезно выполнить нижеследующие упражнения, посвященные анализу поведения решений. После того, как вы научитесь использовать численные процедуры MATLAB, предназначенные для решения ЗГУ, вернитесь к выполнению этих упражнений и выполните соответствующие вычисления.

■ УПРАЖНЕНИЕ 3.5

Покажите, что общее решение ОДУ

$$y''' - 2y'' - y' + 2y = 0$$

имеет вид

$$y(x) = Ae^x + Be^{2x} + Ce^{-x}.$$

Покажите, что при граничных условиях

$$y(0) = 1,\ y(+\infty) = 0,\ y'(+\infty) = 0$$

рассматриваемая задача имеет единственное решение. Покажите, что при граничных условиях

$$y(0) = 1,\ y'(0) = 1,\ y(+\infty) = 0$$

задача не имеет решений. Исследуйте численно при граничных условиях

$$y(0) = 1,\ y'(0) = 1,\ y(20) = 0$$

обусловленность линейной системы, определяющей константы A, B и C.

■ УПРАЖНЕНИЕ 3.6

В работе [Murphy, 1965] обобщены классические результаты о автомодельном решении уравнения Фолкнера–Скана (Falkner–Skan) для пограничного слоя ламинарного потока несжимаемой жидкости, текущей вдоль искривленной поверхности. В качестве математической модели рассматривается ЗГУ, определяемая ОДУ

$$f'''' + (\Omega + f)f''' + \Omega f f'' = \gamma[f'f'' + \Omega(f')^2]$$

с граничными условиями в начале координат

$$f(0) = 0,\ f'(0) = 0$$

и граничными условиями при $x \to \infty$

$$f'(x) \sim e^{-\Omega x},\ f''(x) \sim -\Omega e^{-\Omega x}.$$

Ω — положительный параметр, характеризующий кривизну поверхности и γ — константа, определяемая градиентом давления. Найдите решение ОДУ, характеризующееся при $x \to \infty$ поведением

$$f(x) \sim \delta + \rho e^{-\lambda x}.$$

Эта задача эквивалента нахождению значений констант δ, ρ и $\lambda > 0$, таких что $\delta + \rho e^{-\lambda x}$ удовлетворяет ОДУ при $x \to \infty$. Убедитесь в том, что одним из вариантов решения является $\lambda = \Omega$. Выбрав адекватные значения констант, докажите, что существуют решения, удовлетворяющие граничным условиям на бесконечности, тем самым показав, что граничные условия, приведенные в вышеупомянутой работе, совместимы с указанным поведением решений рассматриваемого ОДУ. Определите *точное* решение ОДУ, удовлетворяющее двум граничным условиям на бесконечности и одному из граничных условий в начале координат. Это решение может быть использовано в качестве критерия выбора решения при численном решении ЗГУ.

■ УПРАЖНЕНИЕ 3.7

В работе [Ames & Lohner, 1981] исследовались модели переноса, реакции и рассеяния загрязнений в реках. Одна из моделей имеет вид системы трех ДУЧП первого порядка относительно одной пространственной переменной x и переменной времени t. Задавшись целью нахождения решений бегущей волны, зависящих только от переменной $z = x - t$, авторы свели систему ДУЧП к системе ОДУ

$$f'' = \beta g f,\ g'' = -\beta g h,\ h'' = \lambda \beta g h,$$

где f — концентрация загрязняющего агента, g — концентрация бактерий и h — концентрация углерода; физические параметры β и λ являются постоянными. Можно показать, что из уравнений для g и h следует

$$g(z) = E - \frac{h(z)}{\lambda},$$

где E — некоторое заданное значение $g(\infty)$. Тогда система ОДУ может быть представлена в следующем виде

$$h'' = \lambda\beta\left(E - \frac{h}{\lambda}\right)h,\ f'' = \beta\left(E - \frac{h}{\lambda}\right)f.$$

Эти уравнения должны быть решены при граничных условиях

$$h(0) = 1,\ f(0) = 1,\ h(\infty) = 0,\ f(\infty) = 0.$$

В вышеуказанной работе рассматривалось множество вариантов, соответствующих различному выбору значений параметров. В частности, было показано, что при $\beta = 10$, $\lambda = 10$ и $E = 1$ верхней и нижней границей значений для $f(0.1)$ являются 0.730 и 0.729 соответственно. Покажите, что при этих значениях параметров $h(z)$ асимптотически кратно e^{-10z} и $f(z)$ кратно $e^{-\sqrt{10}z}$. С учетом этого поведения решений, решите ЗГУ, заменив граничные условия на бесконечности граничными условиями $h(Z) = 0$ и $f(Z) = 0$, где Z — некоторое конечное значение. Для лучшего понимания особенностей поведения полученного решения полезно выполнить численный анализ при различных значениях $Z = 2$, 3 и 4. Если при численном решении ЗГУ возникли трудности, уменьшите значение Z и снова попытайтесь решить задачу. Если численное решение рассматриваемой ЗГУ удалось получить, исследуйте его и убедитесь в том, что оно характеризуется требуемыми свойствами на всем интервале интегрирования. Если решение не обладает желаемыми характеристиками, увеличьте Z и попытайтесь выполнить вычисления заново. В рассматриваемой задаче в качестве критериев выбора решения можно использовать асимптотические выражения для $h(z)$ и $f(z)$, однако в процессе экспериментирования вы убедитесь, что в качестве этих критериев вполне можно использовать постоянные значения. Проверьте, что полученное численное решение для $f(z)$ удовлетворяет предельным значениям, полученным в упомянутой работе.

■ УПРАЖНЕНИЕ 3.8

В работе [Bailey et al., 1968, пример 7.3] рассматривается автомодельное решение для нестационарного потока газа в полубесконечной пористой среде, которая первоначально заполнена равномерно распределенным газом. Соответствующая ЗГУ определяется ОДУ

$$w''(z) + \frac{2z}{\sqrt{1 - \alpha w(z)}}w'(z) = 0$$

с граничными условиями

$$w(0) = 1,\ w(+\infty) = 0.$$

В пример 8.4 вышеупомянутой работы рассматривается численное решение этой задачи при различных значениях параметра $0 < \alpha < 1$. Решите эту ЗНУ при $\alpha = 0.8$. В большинстве примеров этой главы особое внимание уделяется решению ЗГУ на основе анализа поведения решений вблизи сингулярных точек, однако существуют случаи, когда задачи, определенные на бесконечных интервалах, могут быть решены непосредственно. Найдите численное решение рассматриваемой в этом примере задачи, заменив граничное условие на бесконечности на граничное

условие $w(Z) = 0$, где Z — некоторая конечная величина. В целях лучшего понимания свойств решения, выполните численное интегрирование при различных значениях $Z = 2$, 3 и 4. Используемый в численной процедуре критерий выбора решения $w(z)$ должен соответствовать физически обусловленному условию $0 \leqslant w(z) \leqslant 1$. При выполнении этого условия правая часть рассматриваемого ОДУ может быть легко вычислена.

Заметим, что поскольку $w(\infty) = 0$, при больших z соответствующая аппроксимация рассматриваемого ОДУ имеет вид $w''(z)+2zw'(z) = 0$. Решая это уравнение, получаем

$$w'(z) \sim \beta e^{-z^2}.$$

Интегрируя и накладывая граничное условие на бесконечности, получаем

$$w(z) \sim -\beta \int_z^\infty e^{-t^2} dt = \left(-\frac{\beta\sqrt{\pi}}{2}\right) \operatorname{erfc}(z).$$

Из стандартного асимптотического представления дополнительной функции ошибки

$$\operatorname{erfc}(z) \sim \frac{e^{-z^2}}{z\sqrt{\pi}}$$

следует, что $w(z)$ стремится к предельному значению $w(\infty) = 0$ очень быстро. Поэтому при численном интегрировании мы можем наложить граничное условие $w(Z) = 0$, используя при этом относительно малое значение Z. Мы действительно должны использовать малое значение Z, т. к. при больших Z численная процедура не сможет отличить решение $w(z)$ от тождественно нулевого решения ОДУ. При решении некоторых задач иногда возникает необходимость в предоставлении дополнительной информации о поведении решений вблизи сингулярной точки. В нашем примере это может быть сделано следующим образом: константа β рассматривается в качестве неизвестного параметра и вместо $w(Z) = 0$ используются два граничных условия

$$w'(Z) = \beta e^{-Z^2}, \quad w(Z) = \left(-\frac{\beta\sqrt{\pi}}{2}\right) \operatorname{erfc}(Z).$$

Аналитические аппроксимации для $w(z)$ и $w'(z)$ являются точными лишь при больших z, но в численной процедуре они могут использоваться в качестве соответствующих критериев выбора для $w(z)$ и $w'(z)$ при всех z.

■ УПРАЖНЕНИЕ 3.9

Рассмотрим уравнение Томаса–Ферми

$$y'' = x^{-1/2} y^{3/2},$$

которое должно быть решено с граничными условиями

$$y(0) = 1, \quad y(+\infty) = 0.$$

Эта ЗГУ возникает при исследовании полуклассической модели плотности заряда в атомах с большим атомным числом. Сингулярности, имеющие место на обоих

концах, исследовались в работах [Davis, 1962] и [Bender & Orszag, 1999]. В первой работе рассматривалось разложение в ряд решений $y(x)$ при $x \to 0$. Очевидно, что в этих рядах имеются члены с дробными степенями, т. к. с учетом $y(0) = 1$ из ОДУ следует, что $y'' \sim x^{-1/2}$ при $x \to 0$ и, следовательно, в разложении в ряд для $y(x)$ должен быть член вида $\frac{4}{3}x^{3/2}$. Кроме того, должны быть также члены меньшего порядка, такие что выполнено граничное условие в $x = 0$. С учетом всего вышесказанного, решение следует искать в виде

$$y(x) = 1 + px + \frac{4}{3}x^{3/2} + bx^2 + cx^{5/2} + \ldots$$

Дальнейшее изложение может быть упрощено, если переписать уравнение в следующей форме

$$x(y'')^2 = y^3.$$

Покажите, что в приведенном выше выражении для вида решения коэффициент p представляет собой свободный параметр, $b = 0$ и $c = 2p/5$. Во второй из вышеупомянутых работ исследовалось асимптотическое поведение $y(x)$ при $x \to \infty$. Покажите, что в условиях $y(x) \sim ax^{\alpha}$ функция $y_0(x) = 144x^{-3}$ является точным решением ОДУ, удовлетворяющим граничному условию на бесконечности. Для исследования поведения общего решения вблизи этого частного решения запишите $y(x) = y_0(x) + \varepsilon(x)$ и покажите, что если функции $\varepsilon(x)$ малы по сравнению с $y_0(x)$, то

$$y^{3/2}(x) \approx y_0^{3/2}(x) + \tfrac{3}{2}y_0^{1/2}(x)\varepsilon(x)$$

и, следовательно, аппроксимация функции коррекции $\varepsilon(x)$ может быть получена как решение уравнения $\varepsilon''(x) = 18x^{-2}\varepsilon(x)$. Найдите решения $\varepsilon(x)$ в виде cx^{γ}. Исключив из рассмотрения решения, которые не стремятся к нулю при $x \to \infty$, покажите, что решения ОДУ, которые близки к $y_0(x)$ и удовлетворяют граничному условию на бесконечности, имеют вид $y(x) \sim 144x^{-3} + cx^{\gamma}$, где $\gamma = (1 - \sqrt{73})/2 \approx$ ≈ -3.772 и c — произвольная константа.

Переместите граничное условие в начале координат в точку, соответствующую некоторому малому числу $d > 0$, обеспечив при этом равенство величин $y(d)$ и $y'(d)$ соответствующим значениям, вычисленным с использованием аппроксимирующих рядов в предположении, что p рассматривается как неизвестный параметр. Переместите граничное условие на бесконечности в точку, соответствующую некоторому большому числу D, обеспечив при этом равенство величин $y(D)$ и $y_0(D)$. Поскольку решение убывает не как экспонента, а как убывающая рациональная функция, необходимо использовать относительно большое значение D. Решите рассматриваемую ЗГУ при различных значениях d и D, используя заданные по умолчанию значения допустимых ошибок вычислений. В качестве интервалов интегрирования используйте [1, 20], [0.1, 40] и [0.01, 60], а в качестве критериев отбора решения на промежутке $[d, D]$

$$y(x) \approx 1,\ y'(x) \approx 0$$

при $x \in [d, 1]$ и

$$y(x) \approx 144x^{-3},\ y'(x) \approx -432x^{-4}$$

при $x \in (1, D]$. После успешного нахождения решения на одном из интервалов $[d, D]$ можно использовать функцию `bvpinit`, позволяющую экстраполировать это решение на больший интервал. Полученная таким образом экстраполяция может быть использована в качестве критерия отбора решения при выполнении интегрирования уравнения на этом большом интервале. Для вычисления $x^{-1/2}y^{3/2}$ следует использовать программный код

```
max(y(1),0)^(3/2) / sqrt(x)
```

т. к. промежуточные аппроксимации $y(x)$ могут быть отрицательными, что приведет к получению комплексных значений для соответствующих дробных степеней этой величины. Выведите вычисленные на различных интервалах решения на одном графике и проанализируйте полученные результаты. На рисунке 4.10 в работе [Bender & Orszag, 1999] показаны некоторые результаты решения рассматриваемой ЗГУ с использованием метода пристрелки. Дополните результаты ваших расчетов для последнего интервала значением $y(0) = 1$ и выведите график решения на координатной плоскости, заданной функцией `axis[0 15 0 1]`, и сравните полученный результат с вышеупомянутым рисунком. Сравните вычисленное вами значение параметра $p = y'(0)$ со значением $p = -1.588$, полученным в работах [Bender & Orszag, 1999] и [Davis, 1962].

§ 3.4. ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ ЗГУ

Предложенный в параграфе 3.2 теоретический подход к решению задач с граничными условиями (ЗГУ) основан на использовании имеющихся результатов о решении задач с начальными условиями (ЗНУ) и решении нелинейных алгебраических уравнений. Поскольку существуют эффективные программы, предназначенные для решения обеих задач, было бы естественным объединить эти программы в единой программе, предназначенной для решения ЗГУ. Одним из методов, в которых используется это подход, является *метод пристрелки*. Использование уже готовых и проверенных численных методов в качестве компонентов для решения ЗГУ представляется естественным, но читатель возможно удивится, когда узнает, что в наиболее широко используемых численных процедурах решения ЗГУ метод пристрелки *не* используется. Основным недостатком этого метода является то, что при его применении может возникнуть необходимость в интегрировании неустойчивой ЗНУ. В этом случае решение рассматриваемой ЗГУ может быть нечувствительно к изменению граничных значений, в то время как решения соответствующей ЗНУ, возникающей при применении метода пристрелки, чувствительны к изменению начальных значений. Рассмотрим в качестве простейшего примера уравнение

$$y'' - 100y = 0$$

с граничными условиями $y(0) = 1$ и $y(10) = B$. Применяя метод пристрелки (слева направо) мы приходим к необходимости решать ЗНУ с начальными значениями $y(0) = 1$ и $y'(0) = s$. Аналитическое решение этого уравнения имеет вид

$$y(x, s) = \cosh(10x) + 0.1s \sinh(10x).$$

Частная производная $\frac{\partial y}{\partial s} = 0.1\sinh(10x)$ не превышает величины $0.1\sinh(100) \approx \approx 1.3 \cdot 10^{42}$ на интервале $[0, 10]$. Значение s, при котором выполнено граничное условие в $x = 10$, определяется равенством

$$s = \frac{10(B - \cosh(100))}{\sinh(100)}.$$

Подставляя эту величину в общее решение ЗНУ, получаем

$$\left|\frac{\partial y}{\partial B}\right| = \left|\frac{\sinh(10x)}{\sinh(100)}\right| \leqslant 1.$$

Очевидно, что чувствительность решения ЗНУ к изменениям начальнсго значения производной $y'(0) = s$ существенно более выражена, чем чувствительность решения ЗГУ к изменениям граничного значения $y(10) = B$. Если ЗНУ достаточно устойчива, метод пристрелки может быть вполне эффективен. В случае исследования неустойчивых ЗНУ программная реализация метода пристрелки может оказаться недееспособной, поскольку решение может уйти на бесконечность до момента времени, соответствующего концу интервала интегрирования. Даже если конец этого интервала достигается, точность полученного результата часто оказывается неприемлемой. Поскольку решение нелинейных алгебраических уравнений не может быть вычислено абсолютно точно, полученные с использованием соответствующей численной процедуры начальные значения, необходимые для решения ЗГУ, вычисляются лишь с некоторой точностью, что в свою очередь оказывает влияние на точность соответствующего решения ЗГУ. С использованием рассмотренных в этой главе методик решения задач, характеризующихся наличием сингулярных точек и рассматриваемых на бесконечных интервалах, можно обеспечить успешное использование численных процедур, основанных на методе пристрелки. Однако это утверждение верно лишь для устойчивых задач и поэтому соответствующие численные процедуры используются нечасто. Численная процедура `D02SAF` [Gladwell, 1987] и ее модификации, входящие в пакет прикладных программ NAG, представляют собой простейшие реализации, которые находят широкое применение при исследовании большого класса задач с граничными условиями. Пример 3.5.5 иллюстрирует использование этой процедуры.

Одним из путей преодоления сложностей, возникающих при использовании метода пристрелки для решения ЗНУ, является разбиение интервала интегрирования на несколько подинтервалов. В качестве обоснования этой методики напомним (см. главу 2), что оценка области устойчивости ЗНУ, определяемой ОДУ с правой частью $f(t, y)$, удовлетворяющей условию Липшица с константой L, включает член с множителем $e^{L(b-a)}$. С учетом этого результата можно утверждать, что уменьшение длины интервала интегрирования приводит к экспоненциальному увеличению степени устойчивости ЗНУ. После разбиения первоначального интервала, ОДУ интегрируется независимо на последовательности нескольких интервалов. Решения на соседних интервалах должны совпадать в соответствующих точках разбиения. Совокупность приближенных решений на подынтервалах образует непрерывную аппроксимацию решения на всем исходном интервале интегрирования. Граничные условия и условия непрерывности составляют множество нелинейных алгебраических уравнений с $(N + 1)d$ неизвестными, где d — число ОДУ первого порядка

и N — число точек разбиения. Этот подход называется *методом многократной пристрелки*. Эти алгебраические уравнения решаются с использованием различных модификаций метода Ньютона во многом аналогично тому, как это делается при вычислении неявных формул при решении ЗНУ. На каждом шаге итераций решается система линейных уравнений, обладающая специальной структурой, которую очень важно выявить и использовать, поскольку эта система может быть очень большой. Если рассматриваемая ЗНУ настолько неустойчива, что оказывается целесообразным использовать метод многократной пристрелки, для успешного решения системы линейных уравнений необходимо выбрать главный элемент этой матрицы. Множество работ было посвящено вопросу решения линейных систем большой размерности, возникающих при решении ЗГУ с использованием метода многократной пристрелки. При правильном выборе точек разбиения и метода решения соответствующей линейной системы численные процедуры, основанные на методе многократной пристрелки, могут быть вполне эффективны. Основной сложностью, возникающей при написании программ, использующих метод многократной пристрелки, является разработка алгоритма выбора первоначального множества точек разбиения, их перемещения, удаления и добавления. В качестве примеров программ решения ЗГУ, в которых реализован метод многократной пристрелки, можно указать MUSN [Mattheij & Staarink, 1984a,b] и DD04 [England & Reid, *H2KL*].

При такой реализации метода многократной пристрелки на каждом из подынтервалов с использованием эффективных численных процедур решается отдельная ЗНУ. Поскольку эти процедуры в процессе своей работы меняют величину шага интегрирования, количество точек сетки на каждом подынтервале, а также интервалы между ними заранее неизвестны. Существует множество реализаций метода конечных разностей и одна из наиболее популярных может рассматриваться как метод многократной пристрелки, в котором используется одношаговый метод интегрирования в экстремальном случае, когда на интервале между точками разбиения выполняется только один шаг. Эти методы конечных разностей позволяют аппроксимировать решение ОДУ (3.1) последовательностью решений $y_0, y_1, \ldots, y_N$, вычисленных на каждом из подынтервалов $[x_i, x_{i+1}]$ сетки $a = x_0 < x_1 < \ldots < x_N = b$ при помощи одношагового метода. В этом случае граничные условия (3.2) имеют вид $g(y_0, y_N) = 0$. Формула трапеций представляет собой простой и важный пример аппроксимации ОДУ

$$y_{i+1} - y_i = \frac{h_i}{2}[f(x_i, y_i) + f(x_{i+1}, y_{i+1})],$$

где $i = 0, 1, \ldots, N-1$. Эти N уравнений вместе с граничными условиями составляют систему нелинейных алгебраических уравнений относительно $y_i \approx y(x_i)$. При использовании подобного подхода решение оказывается определенным неявно, даже в случае использования явного одношагового метода интегрирования ЗНУ, например метода Эйлера

$$y_{i+1} - y_i = h_i f(x_i, y_i).$$

Это обстоятельство обусловлено фундаментальным различием между ЗНУ и ЗГУ. Для определения решения ЗНУ достаточно задать одну начальную точку, после

чего решение будет *однонаправленно* эволюционировать из этой точки. С другой стороны, при рассмотрении ЗГУ решение определяется двумя или более граничными значениями и направление эволюции решения оказывается неопределенным, и поэтому возникает необходимость в учете информации о вычисленном решении во всех его точках. Поскольку явные одношаговые методы решения ЗНУ фактически становятся при решении ЗГУ неявными, их использование в этом случае оказывается не столь эффективным, как в случае решения ЗНУ. В случае решения ЗГУ трудности вычислительного характера, связанные с использованием неявных одношаговых методов, нивелируются и можно воспользоваться их преимуществами — точностью и устойчивостью. В частности, формула трапеций является формулой второго порядка, которая характеризуется хорошей устойчивостью и может быть одинаково эффективно использована при интегрировании в прямом и обратном времени.

Также как и в случае использования метода многократной пристрелки, применение формулы трапеций приводит к необходимости решения $(N+1)d$ нелинейных уравнений. Однако при этом число N может быть очень большим, и при выполнении вычислений необходимо учитывать структуру этих уравнений, если мы хотим, чтобы вычислительный метод имел практическое значение. Нелинейные уравнения обычно решаются с использованием какой-либо модификации метода Ньютона. Структура линейных систем, которые необходимо решать на каждой итерации метода Ньютона, становится более понятной в предположении, что уравнения и граничные условия являются линейными, т. е.

$$y' = J(x)y + q(x), \tag{3.11}$$

$$B_a y(a) + B_b y(b) = \beta. \tag{3.12}$$

В этом случае формула трапеций на интервале $[x_i, x_{i+1}]$ принимает вид

$$\left[-I - \frac{h_i}{2}J(x_i)\right] y_i + \left[I - \frac{h_i}{2}J(x_{i+1})\right] y_{i+1} = \frac{h_i}{2}[q(x_i) + q(x_{i+1})]. \tag{3.13}$$

После масштабирования эти уравнения могут быть представлены в следующей матричной форме

$$\begin{pmatrix} S_0 & R_0 & & & & \\ & S_1 & R_1 & & & \\ & & \ddots & \ddots & & \\ & & & S_{N-1} & R_{N-1} & \\ B_a & & & & B_b \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \\ y_N \end{pmatrix} = \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{N-1} \\ \beta \end{pmatrix}, \tag{3.14}$$

где

$$S_i = -\frac{2}{h_i}I - J(x_i),\ R_i = \frac{2}{h_i}I - J(x_{i+1}),\ v_i = q(x_i) + q(x_{i+1})$$

при $i = 0, 1, \ldots, N-1$. Одним из способов использования структуры этой матрицы является определение ее как разреженной матрицы общего вида. Это может быть легко сделано в MATLAB, и именно этот способ увеличения эффективности вычислений используется в процедуре `bvp4c`. В большинстве других численных

процедур используются другие подходы к увеличению эффективности вычислений. Например, существуют процедуры, специально предназначенные для решения ЗГУ с разделенными граничными условиями

$$B_a y(a) = \beta_a,\ B_b y(b) = \beta_b.$$

Системы линейных уравнений, возникающие при решении подобных задач, имеют простую структуру.

Если граничные условия являются разделенными, уравнения могут быть переписаны в следующем виде

$$\begin{pmatrix} B_a & & & & & \\ S_0 & R_0 & & & & \\ & S_1 & R_1 & & & \\ & & \ddots & \ddots & & \\ & & & S_{N-1} & R_{N-1} & \\ & & & & B_b & \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{N-1} \\ y_N \end{pmatrix} = \begin{pmatrix} \beta_a \\ v_0 \\ v_1 \\ \vdots \\ v_{N-1} \\ \beta_b \end{pmatrix}.$$

Очевидно, что подобную матрицу удобно использовать и хранить в виде ленточной матрицы. Алгоритм исключения Гаусса с выбором ведущего элемента строки является эффективным методом решения линейных ленточных уравнений. В более сложных и эффективных схемах вычислений с такими матрицами учитывается их почти блочно-диагональная структура, а также применяется выбор ведущего элемента по всей матрице для минимизации требуемого объема оперативной памяти компьютера и увеличения общей устойчивости вычислительного процесса. Обзор этих схем приведен в работе [Amodio et al., 2000]. Следует отметить, что структура представленной выше матрицы не такая регулярная, как могло бы показаться с первого взгляда: число строк подматриц B_a и B_b не равно d, т. е. они не являются квадратными. Это связано с тем, что в случае, когда граничные условия являются разделенными, число строк в подматрице B_a равно числу граничных условий, заданных в граничной точке a (аналогичное утверждение верно для B_b); эти две подматрицы имеют в совокупности d ненулевых строки, которые соответствуют граничным условиям. Одной из причин того, что столь ограничительная вычислительная схема широко используется в многочисленных численных процедурах, является то обстоятельство, что в отличие от случая использования разреженной матрицы общего вида, можно заранее предсказать и выделить объем оперативной памяти, необходимой для решения соответствующих линейных систем.

Формула трапеций, используемая как одношаговый метод, имеет порядок 2, т. е. решение $y(x)$ удовлетворяет формуле (3.13) с локальной погрешностью τ_i в x_i порядка $O(h_i^3)$. Определим ошибку равенством $e_i = y(x_i) - y_i$. Тогда после подстановки y_i и $y(x_i)$ в (3.14) и вычитания полученных равенств, получаем

$$\begin{pmatrix} S_0 & R_0 & & & & \\ & S_1 & R_1 & & & \\ & & \ddots & \ddots & & \\ & & & S_{N-1} & R_{N-1} & \\ B_a & & & & B_b & \end{pmatrix} \begin{pmatrix} e_0 \\ e_1 \\ \vdots \\ e_{N-1} \\ e_N \end{pmatrix} = \begin{pmatrix} \sigma_0 \\ \sigma_1 \\ \vdots \\ \sigma_{N-1} \\ 0 \end{pmatrix}, \qquad (3.15)$$

где векторы σ_i представляют собой масштабированные величины погрешностей $(2/h_i)\tau_i$. Заметим, что в граничных точках локальные погрешности формул равны нулю. Таким образом, вектор ошибок в точках сетки равен соответствующей обратной матрице, умноженной на вектор масштабированных погрешностей. Метод конечных разностей называется устойчивым, если норма этой обратной матрицы существует и *равномерно* ограничена. При $h = \max_i h_i$ масштабированные погрешности формул имеют порядок $O(h^2)$, и можно заключить, что метод конечных разностей, основанный на формуле трапеций, является сходящимся методом второго порядка, если он является устойчивым. Наиболее трудным этапом доказательства сходимости метода является установление факта устойчивости метода конечных разностей. Можно показать, что если ЗГУ, определяемая ОДУ (3.11) и граничными условиями (3.12), имеет единственное решение и если правая часть ОДУ является достаточно гладкой функцией, то устойчивость схемы конечных элементов следует из устойчивости одношагового метода, используемого при решении ЗНУ. Далее, скорость сходимости схемы конечных элементов для ЗГУ равна скорости сходимости одношагового метода для ЗНУ. Доказательство этого общего результата имеет чисто технический характер, и мы не будем его здесь приводить; более детальная информация по этому вопросу изложена в специальных источниках, например в работах [Ascher et al., 1995] и [Keller, 1992]. По аналогичным причинам нами была рассмотрена лишь простейшая ситуация, когда ЗГУ определяется линейными ОДУ и линейными граничными условиями. Этого вполне достаточно для того, чтобы прояснить существо рассматриваемых проблем; более детальная информация, касающаяся модификации доказательства для нелинейного случая, может быть найдена в специальной литературе.

Многие задачи могут быть решены более эффективно с использованием методов, порядок которых превышает 2. При выполнении одного шага длиной h_i из точки x_i с использованием метода Рунге–Кутта с s стадиями аппроксимации вычисляется s промежуточных значений $y_{i,j} \approx y(x_{i,j})$ в точках $x_{i,j} = x_i + \alpha_j h_i$. В главе 2 мы рассматривали явные РК-методы, в которых эти значения вычислялись непосредственно, но в общем случае они определяются как решения системы s нелинейных алгебраических уравнений

$$y_{i,j} = y_i + h_i \sum_{k=1}^{s} \beta_{j,k} f(x_{i,k}, y_{i,k}). \tag{3.16}$$

В этом случае новое приближенное значение может быть вычислено в соответствии с равенством

$$y_{i+1} = y_i + h_i \sum_{j=1}^{s} \gamma_k f(x_{i,j}, y_{i,j}).$$

Неявные формулы Рунге–Кутта (НРК-формулы), которые при нахождении решения $y' = f(x)$ сводятся к квадратурным формулам Гаусса, находят широкое применение. Это связано с тем, что эти формулы обладают великолепной устойчивостью и среди методов с s стадиями аппроксимации имеют наибольший порядок $2s$. При рассмотрении некоторых более простых формул удобно ввести обозначение $x_{i+1/2}$ для средней точки интервала $[x_i, x_{i+1}]$, а также обозначение $y_{i+1/2}$ для соответствующей промежуточной аппроксимации. Тогда в качестве примера формулы

Таблица 3.1: Формула Симпсона порядка 4

0	0	0	0
$\frac{1}{2}$	$\frac{5}{24}$	$\frac{1}{3}$	$-\frac{1}{24}$
1	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$
	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$

Гаусса наименьшего порядка можно рассмотреть формулу прямоугольников

$$y_{i+1} - y_i = h_i f(x_{i+1/2}, y_{i+1/2}),$$

имеющую порядок 2. Формулы Гаусса не вычисляются в концах подынтервалов. В подпараграфе 3.3.1 было отмечено, что это обстоятельство может оказаться полезным в случаях решения задач с сингулярностью на конце интервала интегрирования.

Другой класс НРК-формул сводится при решении $y' = f(x)$ к квадратурным формулам Лобатто. В качестве примера подобной формулы наименьшего порядка можно рассмотреть формулу трапеций. Формула Лобатто следующего порядка называется формулой Симпсона, т. к. она сводится к квадратурной формуле Симпсона. В таблице 3.1 приведены коэффициенты формулы более общего вида. С использованием этой таблицы можно выписать вычислительные формулы, если в качестве первой стадии аппроксимации используется значение y_i, а в качестве последней — y_{i+1}

$$y_{i+1/2} = y_i + h_i \left[\frac{5}{24} f(x_i, y_i) + \frac{1}{3} f(x_{i+1/2}, y_{i+1/2}) - \frac{1}{24} f(x_{i+1}, y_{i+1}) \right],$$

$$y_{i+1} = y_i + h_i \left[\frac{1}{6} f(x_i, y_i) + \frac{2}{3} f(x_{i+1/2}, y_{i+1/2}) + \frac{1}{6} f(x_{i+1}, y_{i+1}) \right].$$

Нестрого говоря, все сформулированные результаты для формулы трапеций, остаются применимыми и в отношении методов конечных разностей, основанных на формуле Рунге–Кутта. Однако при программной реализации этих формул возникают определенные трудности. При написании эффективного программного кода для вычисления формул высокого порядка используется *метод отложенной коррекции*, заключающийся в том, что эти формулы вычисляются в виде коррекций некоторых других формул, вычисление которых может быть выполнено сравнительно легко. Программа PASVA3 [Lentini & Pereyra, 1974], была первой робастной программой, предназначенной для решения ЗГУ и основанной на использовании метода отложенной коррекции. В этой программе формула трапеций использовалась в качестве основной. Вычисление формул более высокого порядка осуществлялось с использованием последовательно определяемых аппроксимирующих членов в разложении погрешности формулы трапеций. В работе [Cash & Wright, 1991] были

разработаны несколько эффективных численных процедур решения ЗНУ, среди которых хотелось бы отметить в первую очередь TWPBVP. Эта реализация, основанная на использовании метода отложенной коррекции, позволила получить результаты, сопоставимые с результатами использования (неявной) формулы Рунге–Кутта и других формул более высокого порядка. В этой программе в качестве основной использовалась формула Симпсона.

В других широко используемых численных процедурах решения ЗНУ используются непосредственно НРК-методы порядка больше 2. Основная трудность, связанная с реализацией этих программ, заключается в вычислении промежуточных аппроксимаций $y_{i,j}$. В соответствии с определением этих величин, они могут быть вычислены с использованием лишь значений y_i и y_{i+1}. Исключение промежуточных аппроксимаций из вычислительных формул называется *конденсацией*. Выполнение этой процедуры имеет важное практическое значение, поскольку позволяет существенно снизить размерность задачи. В случае использования формулы Симпсона конденсация может быть выполнена аналитически

$$y_{i+1} = y_i + \frac{h_i}{6}\left[f(x_i, y_i) + f(x_{i+1}, y_{i+1}) + \right.$$
$$\left. + 4f\left(x_{i+1/2}, \frac{y_{i+1} + y_i}{2} + \frac{h_i}{8}[f(x_i, y_i) - f(x_{i+1}, y_{i+1})]\right)\right]. \quad (3.17)$$

В численной процедуре bvp4c используется формула Симпсона с аналитической конденсацией. В общем случае конденсация выполняется численно. Для того, чтобы понять что при этом происходит, рассмотрим линейное ОДУ с правой частью $J(x)y + g(x)$. При подстановке этой функции в (3.16) вместо $f(x, y)$ возникает система линейных уравнений относительно промежуточных стадий аппроксимации $y_{i,j}$, зависящая только от y_i и y_{i+1}. Эта подсистема может быть легко решена, в результате чего соответствующие неизвестные могут быть исключены из всей системы. Будучи примененными к системам первого порядка, численные процедуры COLSYS [Ascher, Christiansen & Russell, 1979, 1981] и COLNEW [Ascher et al., 1995; Bader & Ascher, 1987] могут рассматриваться как реализации семейства НРК-методов, сводящихся к квадратурным формулам Гаусса. В этих численных процедурах промежуточные значения исключаются численно.

Методы конечных разностей позволяют определить решение только в узлах сетки. При решении ЗНУ с использованием явных методов Рунге–Кутта нахождение приближенного решения в других точках вызывает определенные трудности. Для некоторых формул Рунге–Кутта были разработаны непрерывные обобщения, позволяющие получить решение в промежуточных точках шага интегрирования. При естественном непрерывном обобщении НРК-метода с s стадиями аппроксимации полином $S(x)$ определяется интерполяционными условиями $S(x_i) = y_i$ и $S'(x_{i,j}) = f(x_{i,j}, y_{i,j})$, $j = 1, 2, \ldots, s$. Для класса НРК-методов, включая те, что основаны на квадратурных формулах Гаусса, нетрудно показать, что этот полином удовлетворяет $S(x_{i+1}) = y_{i+1}$. Из этого результата следует, что введенное выше естественное непрерывное обобщение действительно является непрерывным, т. е. $S(x) \in C[a, b]$. Можно также показать, что $S(x_{i,j}) = y_{i,j}$. Из этих равенств с учетом интерполяционных условий следует, что $S(x)$ удовлетворяет ОДУ в каждой

промежуточной точке: $S'(x_{i,j}) = f(x_{i,j}, S(x_{i,j}))$. Это свойство называется *коллокацией*. Мы не рассматривали в главе 2 эти естественные непрерывные обобщения, поскольку в общем случае их порядок точности не совпадает с точностью соответствующей формулы. Действительно, для формул из класса квадратурных формул Гаусса точность непрерывных обобщений в два раза меньше точности исходной формулы. В качестве простейшего примера рассмотрим формулу прямоугольника. На подынтервале $[x_i, x_{i+1}]$ непрерывное обобщение $S(x)$ представляет собой линейный полином с $S(x_i) = y_i$ и $S'(x_{i+1/2}) = f(x_{i+1/2}, y_{i+1/2})$. Согласно формуле прямоугольника

$$f(x_{i+1/2}, y_{i+1/2}) = \frac{y_{i+1} - y_i}{h_i}.$$

Поскольку линейный полином $S(x)$ имеет постоянную производную, из приведенного выше выражения для производной $S(x)$ в средней точке следует, что $S(x)$ представляет собой прямую линию, интерполирующую численное решение по обоим концам интервала. Это непрерывное обобщение $S(x)$ принадлежит классу $C[a,b]$, но совершенно очевидно, что оно не принадлежит классу $C^1[a,b]$. Между узлами сетки его порядок точности равен 1.

В общем случае естественное непрерывное обобщение принадлежит лишь классу $C[a,b]$, но для формул из класса квадратурных формул Лобатто в обоих концах подынтервала имеет место свойство коллокации, т. е.

$$S'(x_i) = f(x_i, S(x_i)) = f(x_i, y_i),\ S'(x_{i+1}) = f(x_{i+1}, S(x_{i+1})) = f(x_{i+1}, y_{i+1}).$$

Из этих равенств следует, что для указанных формул выполнено $S(x) \in C^1[a,b]$. Кроме того, естественное непрерывное обобщение для формулы Симпсона имеет тот же порядок точности, что и сама формула. В контексте использования формулы Симпсона в системе MATLAB в качестве основы для численного метода решения ЗНУ она обладает рядом преимуществ. Действительно, полученное численное решение равномерно имеет точность четвертого порядка и принадлежит к классу $C^1[a,b]$ и функции с такими свойствами представляются наиболее удобными для графического исследования средствами MATLAB. Кроме того, предложенная схема вычислительно эффективна, т. к. при аналитической реализации процедуры конденсации вычисление формул не представляет труда.

Классический подход к задаче решения ЗГУ заключается в нахождении функции приближенного решения $S(x)$, зависящей от параметров, которые определяются из требования выполнения граничных условий для $S(x)$, и последующем выполнении процедуры коллокации ОДУ в достаточно большом числе точек. Ранее было показано, что некоторые важные классы формул конечных разностей могут рассматриваться как результат выполнения коллокации ОДУ при непрерывной кусочно-полиномиальной функции $S(x)$ (сплайн). Именно этот подход используется в большинстве численных процедур решения ЗГУ; в частности, это объясняет тот факт, что численная процедура `bvp4c` часто рассматривается как программная реализация методики коллокации. Решение ЗГУ вида (3.1) и (3.2) с использованием `bvp4c` осуществляется путем нахождения непрерывной функции $S(x)$, представляющей собой кубический полином на каждом подынтервале $[x_i, x_{i+1}]$ сетки $a = x_0 < x_1 < \ldots < x_N = b$. Коэффициенты этого кубического полинома

определяются с учетом требования непрерывности $S(x)$ на интервале $[a,b]$, из граничных условий

$$g(S(a), S(b)) = 0,$$

а также из условия выполнения для $S(x)$ рассматриваемого ОДУ в обеих конечных точках и в средней точке каждого из подынтервалов

$$\begin{aligned} S'(x_i) &= f(x_i, S(x_i)), \\ S'(x_{i+1/2}) &= f(x_{i+1/2}, S(x_{i+1/2})), \\ S'(x_{i+1}) &= f(x_{i+1}, S(x_{i+1})). \end{aligned}$$

Как было отмечено выше, из условия коллокации на концах подынтервала следует, что $S(x) \in C^1[a,b]$. В своей совокупности эти условия составляют систему нелинейных алгебраических уравнений относительно коэффициентов кубических полиномов, определяющих функцию $S(x)$. Более детальный анализ показывает, что эта функция $S(x)$ представляет собой естественное непрерывное обобщение формулы Симпсона. Поэтому мы можем одновременно рассматривать этот метод и как метод коллокации, и как метод конечных разностей с непрерывным обобщением. В работе [Enright & Muir, 1996] рассматриваются несколько НРК-формул с непрерывными обобщениями, а также предложена соответствующая программная реализация MIRKDC на языке Fortran. В число рассмотренных в этой работе формул входит формула Симпсона, но следует отметить, что непрерывное обобщение для этой формулы представляет собой полином, степень которого (и, соответственно, точность формулы) больше степени полинома, используемого при естественном непрерывном обобщении в численной процедуре bvp4c. Очевидно, что подход коллокации применим не только к системам ОДУ первого порядка и это обстоятельство представляется весьма важным с учетом того, что непосредственное рассмотрение уравнений высокого порядка (без преобразования к системе уравнений первого порядка) иногда бывает более предпочтительным методом решения задачи. В частности, программы COLSYS и COLNEW, упомянутые выше в связи с методами конечных разностей, отличаются тем, что позволяют решить уравнения высокого порядка непосредственно. В случае, когда эти программы применяются в отношении систем ОДУ первого порядка, лежащие в их основе численные методы могут рассматриваться как эквивалент схемы конечных разностей, основанной на НРК-формулах из класса квадратурных формул Гаусса.

Поскольку ЗГУ могут иметь более одного решения, необходимо задавать критерии, позволяющие численной процедуре отобрать решение, которое интересует пользователя. Во всех используемых на практике численных процедурах численное решение нелинейной ЗГУ выполняется явно или неявно с использованием линеаризации. В этих численных процедурах применяются специальные вычислительные приемы, позволяющие ускорить скорость сходимости, но для обеспечения этой сходимости может оказаться необходимой хорошая первоначальная оценка. В качестве подобных оценок могут выступать либо предсказанные значения решения в заданных узлах сетки, либо функция, позволяющая вычислить эти значения. После того, как обеспечена сходимость для заданной сетки, в численных процедурах осуществляется ее модификация, позволяющая получить точное численное

решение в умеренном количестве узлов сетки. В целом, следует признать, что процедура решения ЗГУ несколько сложнее, чем процедура решения ЗНУ. Наиболее трудной частью задачи решения ЗНУ является первоначальная подстройка величины шага интегрирования на первом шаге, поскольку на каждом из последующих шагов эта подстройка выполняется не более одного раза, причем величина изменения шага невелика. В случае решения ЗГУ наиболее трудной частью задачи является определение первоначальной аппроксимации решения. В значительной мере это бремя возложено на пользователя, который должен задать оценки для сетки и соответствующие значения решения, гарантирующие сходимость. Далее будет кратко рассмотрен вопрос о том, как численные процедуры оценивают величину ошибки и управляют ею.

Наиболее естественный подход к управлению величиной ошибки заключается в том, чтобы оценить погрешности формулы и соответствующим образом выполнить подстройку сетки. Если погрешность формулы на подынтервале $[x_i, x_{i+1}]$ может быть выражена в терминах производной решения, эта производная может быть аппроксимирована путем интерполяции по значениям y_i и y_{i+1} (а также с использованием некоторых других аппроксимаций, полученных на соседних интервалах) с последующим дифференцированием полученного интерполяционного полинома. В главе 2 мы рассматривали аналогичную процедуру аппроксимации погрешности для МДН-формулы первого порядка. В случае применения этой методики для решения ЗГУ важным отличием является то, что имеется возможность использовать значения на обоих концах подынтервала. Погрешность для формулы трапеций вычисляется в программе PASVA3 [Lentini & Pereyra] с использованием именно этого подхода. В действительности эта методика может быть применена и в численных процедурах, основанных на формулах более высокого порядка и допускающих отложенную коррекцию ошибки. Оценки погрешностей для НРК-формул из класса квадратурных формул Гаусса могут быть вычислены с использованием аналогичного метода и этот результат применяется в численных процедурах COLSYS и COLNEW. Другой способ оценки погрешности, рассмотренный нами ранее в главе 2, заключается в сравнении результата применения формулы с результатом применения формулы более высокого порядка. В численной процедуре TWPBVP [Cash & Wright] этот метод используется для выполнения отложенной коррекции, причем это реализовано очень эффективно, поскольку значение формулы вычисляется в виде коррекции значения формулы меньшего порядка.

При наличии оценок погрешности формулы шаг интегрирования может быть подстроен во многом аналогично тому, как это делается при решении ЗНУ. Очевидным отличием является то обстоятельство, что при решении ЗГУ изменяется вся сетка, в то время как при решении ЗНУ на каждом шаге интегрирования изменяется только текущее значение шага. С учетом своей природы задачи с граничными условиями всегда решаются в глобальной постановке. Поэтому изменение сетки в некоторой области, обусловленное большой величиной погрешности формулы, приводит к изменению решения на всем интервале. Вполне возможна ситуация, когда увеличение точности решения в одной области приводит к уменьшению его точности в другой области. В случае использования методов, для которых погрешность формулы на $[x_i, x_{i+1}]$ зависит от производной решения на этом подынтервале, локальное изменение сетки приводит к локальному изменению решения (по крайней

мере с точностью до старших членов разложения), т. е. локальное изменение сетки в целях уменьшения погрешностей формулы в отдельной области в действительности приводит к увеличению точности вычисления всего решения.

Одной из основных трудностей реализации численных процедур решения ЗГУ является то, что используемые схемы для аппроксимации погрешностей формул и соответствующей подстройки сетки оказываются применимыми лишь при достаточно мелкой сетке. Поэтому важно, чтобы полученный численной процедурой результат модификации сетки был правдоподобным даже в случае, когда оценки погрешностей формулы оказались не очень хорошими, что часто случается, если первоначальная сетка слишком разрежена или неадекватна поведению решения. Для увеличения достоверности получаемых результатов в численных процедурах COLSYS и COLNEW предусмотрено дополнительное средство управления величиной погрешности формулы, основанное на другом методе ее оценки. После того, как точность найденного решения стала соответствовать погрешности формулы, интервалы сетки делятся пополам и вычисляется новое решение. Ошибка вычисления этого решения оценивается с использованием процедуры *экстраполяции*, существо которой заключается в следующем. Предположим, что мы имеем возможность вычислить значение функции $F(x;h)$, зависящей от параметра h, и нам необходимо найти $F(x;0)$, предельное значение этой функции при $h \to 0$. Если нам известно, что ошибка $e(x;h) = F(x;h) - F(x;0)$ ведет себя как функция $\phi(x)h^p$ при $h \to 0$, то для получения оценки ошибки более точного результата можно использовать аппроксимации $F(x;h)$ и $F(x;h/2)$, вычисленные при h и $h/2$. Из предположения о характере зависимости величины ошибки от параметра h следует, что

$$e\left(x;\frac{h}{2}\right) \sim \phi(x)h^p 2^{-p}$$

и, следовательно,

$$F(x;h) - F\left(x;\frac{h}{2}\right) = e(x;h) - e\left(x;\frac{h}{2}\right) \sim \phi(x)h^p[1-2^{-p}].$$

Разрешив это уравнение относительно $\phi(x)h^p$, получаем формулу для вычисления оценки ошибки для более точного результата

$$e\left(x;\frac{h}{2}\right) \sim \frac{1}{2^p-1}\left[F(x;h) - F\left(x;\frac{h}{2}\right)\right]. \tag{3.18}$$

В применении изложенного подхода к решению ЗГУ величина h представляет собой максимальный шаг интегрирования, а p — порядок точности метода. Если функция $f(x,y)$ в правой части ОДУ является достаточно гладкой и если сетка настолько мелкая, что в выражении для ошибки старший член доминирует, представленная методика экстраполяции позволяет оценить ошибку и с использованием этого результата доказать, что полученное решение вычислено с требуемой точностью.

Численные процедуры MIRKDC [Enright & Muir] и bvp4c из MATLAB используют необычный метод управления ошибкой, который позволяет повысить их робастность в условиях, когда заданные пользователем оценки решения и сетки

неточны. Эти процедуры позволяют получить приближенные решения из класса $S(x) \in C^1[a,b]$. В этом случае невязка для ОДУ определяется равенством

$$r(x) = S'(x) - f(x, S(x)).$$

Аналогично, невязка для граничных условий имеет вид $\delta = g(S(a), S(b))$. С другой стороны, аппроксимация $S(x)$ представляет собой *точное* решение ЗГУ

$$Y' = f(x,Y) + r(x),\ g(Y(a), Y(b)) - \delta = 0.$$

В контексте метода *обратного анализа ошибок*, $S(x)$ представляет собой верное решение, если оно является точным решением ЗГУ, т.е. «близко» к заданному решению в том смысле, что возмущения $r(x)$ и δ «малы». Если ЗГУ корректно поставлена, малые изменения условий этой задачи должны привести к малым изменениям решения, и поэтому решение, являющееся верным в смысле обратного анализа ошибок, является также верным в обычном смысле, т.е. приближенное решение близко к истинному решению. В численных процедурах `MIRKDC` и `bvp4c` контроль величины ошибки осуществляется по значениям указанных невязок. Этот подход обладает определенными преимуществами, поскольку невязки являются хорошо определенными величинами вне зависимости от степени разреженности сетки. Более того, невязка $r(x)$ может быть вычислена в любой точке x и поэтому мы можем оценить ее величину настолько точно, насколько это представляется нам необходимым (с учетом вычислительных затрат). В численной процедуре `bvp4c` в целях повышения робастности в качестве меры невязки на каждом подынтервале $[x_i, x_{i+1}]$ выступает оценка интеграла

$$\int_{x_i}^{x_{i+1}} \|r(x)\|^2 dx.$$

Приближенное значение этого интеграла вычисляется по пятиточечной квадратурной формуле Лобатто. С учетом того, что невязка обращается в ноль на обоих концах и в средней точке интервала, для выполнения вычислений по этой квадратурной формуле необходимо вычислить лишь два дополнительных значения $r(x)$, т.е. дважды вычислить $f(x, S(x))$. Эта формула позволяет получить правдоподобную оценку величины невязки для любой сетки, и она является асимптотически корректной при $h_i \to 0$. Формула Симпсона с естественным непрерывным обобщением обладает тем свойством, что локальные изменения сетки приводят к локальным изменениям величины невязки (по крайней мере с точностью до старшего члена). Доказательства этих теоретических результатов могут быть найдены в работах [Kierzenka, 1998] и [Kierzenka & Shampine, 2001].

Ранее нами было вскользь отмечено, что метод пристрелки может интерпретироваться как управление величинами невязок. На каждом шаге численная процедура решения ЗНУ управляет величиной локальной ошибки, что эквивалентно управлению величиной невязки для соответствующего непрерывного обобщения численного метода, на котором основана эта численная процедура. Из возникающей системы нелинейных алгебраических уравнений находятся начальные условия, при которых оказываются выполненными граничные условия. Таким образом,

с использованием численной процедуры решения нелинейного уравнения определяются начальные значения, при которых численное решение имеет малую невязку по граничным условиям. В программе, реализующей метод пристрелки, с использованием процедуры решения ЗНУ и процедуры решения системы нелинейных уравнений получается численное решение, удовлетворяющее граничным условиям и ОДУ с малыми невязками. Аналогичная схема может быть предложена в качестве соответствующей интерпретации метода многократной пристрелки.

■ УПРАЖНЕНИЕ 3.10

Покажите, что ЗГУ

$$y'' + 100y = 0$$

с граничными условиями $y(0) = 1$ и $y(10) = B$ нечувствительна к малым изменениям значения B.

■ УПРАЖНЕНИЕ 3.11

С использованием разложения в ряд Тейлора найдите выражение для масштабированных погрешностей в уравнении (3.15) вида $\sigma_i = Kh^p y^{(p+1)}(\eta_i)$. Иными словами, найдите целое число p и константу K.

■ УПРАЖНЕНИЕ 3.12

Докажите справедливость формулы Симпсона с аналитической конденсацией (3.17).

■ УПРАЖНЕНИЕ 3.13

Выпишите систему линейных алгебраических уравнений, возникающую при решении ЗГУ с использованием формулы Симпсона с аналитической конденсацией (3.17) в случае, когда эта ЗГУ определяется линейным ОДУ (3.11) с граничными условиями (3.12).

■ УПРАЖНЕНИЕ 3.14

Покажите, что используемая в численной процедуре `bvp4c` функция $S(x)$, определенная на странице 192, является непрерывно дифференцируемой на интервале $[a, b]$, т. е. $S(x) \in C^1[a, b]$.

§ 3.5. РЕШЕНИЕ ЗГУ В MATLAB

В этом параграфе будут рассмотрены несколько примеров, иллюстрирующих использование в MATLAB процедуры `bvp4c` при численном решении ЗГУ. Изначально задачи с граничными условиями могут иметь различную формулировку, и поэтому часто бывает необходимо выполнить некоторые предварительные действия, направленные на то, чтобы представить эти задачи в форме, пригодной для использования стандартного программного обеспечения. В приведенных ниже примерах показывается, как это может быть сделано для общепринятых формулировок ЗГУ и наиболее широко используемых численных процедур решения ЗГУ. Другой класс примеров, рассмотренных в этой главе, иллюстрирует методики увеличения скорости вычислений, выполняемых с использованием численной процедуры `bvp4c`.

ПРИМЕР 3.5.1

Уравнение Брату представляет собой модель воспламенения топлива и с математической точки зрения является интересным примером бифуркации решений дифференциального уравнения. Анализ этих решений может быть выполнен в значительной мере аналитически [Davis, 1962]. Дифференциальное уравнение имеет вид

$$y'' + \lambda e^y = 0$$

и рассматривается с граничными условиями $y(0) = 0 = y(1)$. Брату показал, что при значениях параметра λ, удовлетворяющих $0 \leqslant \lambda < \lambda^* = 3.51383\ldots$, существует два решения, графики которых имеют вид парабол, вогнутых вниз. При $\lambda \to \lambda^*$ эти два решения сближаются и при $\lambda = \lambda^*$ сливаются в единое решение. При $\lambda > \lambda^*$ решений не существует. В качестве иллюстрации использования численной процедуры `bvp4c`, решим эту ЗГУ при $\lambda = 1$. Это может быть сделано во многом аналогично тому, как это делалось ранее при решении соответствующей ЗНУ, и поэтому мы приведем текст программы и обсудим ее отличительные детали.

```
function sol = ch3ex1
solinit = bvpinit(linspace(0,1,5),@guess);
sol = bvp4c(@odes,@bcs,solinit);
plot(sol.x,sol.y(1,:));
figure
xint = linspace(0,1,100);
Sxint = deval(sol,xint);
plot(xint,Sxint(1,:));

%==============================================
function v = guess(x)
v = [ x*(1-x); 1-2*x ];

function dydx = odes(x,y)
dydx = [ y(2); -exp(y(1)) ];

function res = bcs(ya,yb)
res = [ ya(1); yb(1) ];
```

В простейшем случае процедура `bvp4c` имеет только три входных аргумента: функция `odes` для вычисления ОДУ, функция `bcs` для вычисления невязки по граничным условиям и структура `solinit`, задающая первоначальные оценки сетки и значений решения в точках этой сетки. Функция для вычисления ОДУ полностью аналогична той, что используется в численных процедурах MATLAB, предназначенных для решения ЗНУ. Граничные условия $g(y(a), y(b)) = 0$ задаются следующим образом: при вызове функции `bcs` с текущими значениями аппроксимаций $ya \approx y(a)$ и $yb \approx y(b)$ в качестве входных аргументов, вычисленное значение невязки $g(ya, yb)$ (вектор-столбец) возвращается в виде выходного аргумента этой функции.

При использовании численных процедур решения ЗГУ пользователь должен задать оценку решения, которая является для численной процедуры критерием

отбора искомого решения и облегчающую задачу его вычисления. Эта оценка передается в численную поцедуру `bvp4c` в виде структуры, формируемой с использованием вспомогательной функции `bvpinit`. Первым аргументом функции `bvpinit` является первоначальная оценка сетки, соответствующая предполагаемому характеру поведения решения. В рассматриваемом случае в качестве сетки используются пять точек, равноразнесенных на отрезке $[0, 1]$. Вторым аргументом является первоначальная оценка значений решения в точках заданной сетки. Искомое решение имеет две компоненты $y(x)$ и $y'(x)$. Его оценка может быть задана двумя способами. В представленной программе она задается с использованием отдельной процедуры. Функция $x(1-x)$ в качестве оценки для компоненты $y(x)$ имеет подходящую форму и удовлетворяет граничным условиям. Производная этой функции-оценки для компоненты $y(x)$ используется в качестве оценки для второй компоненты решения $y'(x)$. Если в качестве оценок компонент решения используются постоянные функции, эти оценки могут быть непосредственно переданы в процедуру в виде вектора соответствующих значений. Это удобно и часто приводит к успешному получению решения. Например, при решении рассматриваемой ЗГУ при вызове процедуры `bvpinit` вместо функции `guess` можно было бы использовать вектор `[0.5; 0]`.

Выходным аргументом численной процедуры `bvp4c` является структура-решение, для которой в рассматриваемом примере выбран идентификатор `sol`. Напомним, что получение результата в этой форме является опциональной возможностью для численных процедур решения ЗНУ, но для `bvp4c` эта форма представления выходных данных является единственной. Поле `sol.x` содержит определенные численной процедурой точки сетки, а поле `sol.y` — соответствующие значения решения в этих точках. На рисунке 3.2 показан график решения, полученный с использованием функции `plot`, аргументами которой являлись данные, содержащиеся в этих полях. Вычислительные затраты при решении ЗГУ в значительной мере зависят от числа точек сетки и поэтому в численной процедуре `bvp4c` для получения решения используется лишь столько точек, сколько необходимо для получения достаточно точного решения. Рассматриваемая ЗГУ с уравнением Брату настолько проста, что для получения достаточно точного решения потребовалось использовать лишь пять точек, определенных пользователем путем задания первоначальной оценки сетки. Представленный график построен по точно вычисленным пяти точкам, но для получения гладкой кривой потребовалось аппроксимировать решение в дополнительных точках. Полученное с использованием `bvp4c` решение $S(x) \in C^1[0, 1]$ возвращается в виде структуры-решения `sol`, содержащей всю необходимую информацию для вычисления и графического отображения приближенного решения в промежуточных точках. Так же как и в случае численного решения ЗНУ, эта задача может быть выполнена с использованием дополнительной функции `deval`. Напомним, что первым аргументом этой функции является структура-решение, а вторым аргументом — массив точек, в которых необходимо вычислить решение. При получении изображенного на рисунке 3.3 гладкого графика потребовалось вычислить решение $S(x)$ в 100 равноразнесенных точках. Вычисление кусочно-кубической аппроксимации в функции `deval` векторизовано и поэтому получение сравнительно большого числа значений решения вычислительно малозатратно.

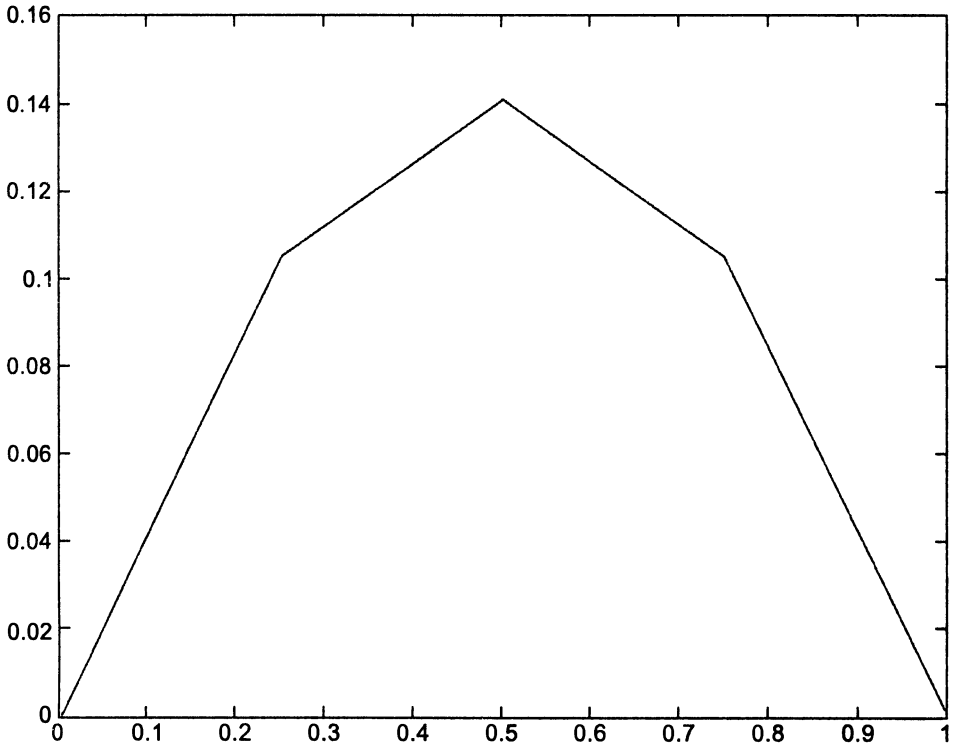

Рис. 3.2. График решения уравнения Брату, построенный лишь по точкам сетки.

Задача Брату имеет два решения. Если в тексте программы `ch3ex1.m` изменить первоначальные оценки решения и его производной, умножив соответствующие величины на коэффициент 5, вы получите решение, аналогичное тому, что изображено на рисунке 3.3. Однако если в качестве этого коэффициента использовать 20, полученное решение будет иным — его максимальное значение увеличится до величины, приблизительно равной 4.1. При значении коэффициента 100 численная процедура не сможет получить решение. Из этих наблюдений следует, что характеристики решения и даже сам факт его успешного вычисления в значительной мере зависят от первоначальных оценок, задаваемых пользователем. В упражнении 3.15 читателю предлагается исследовать закон сохранения, выполняющийся вдоль решений этой задачи. Задача об артиллерийском снаряде, рассматриваемая в упражнении 3.16, во многом сходна с задачей Брату.

ПРИМЕР 3.5.2

В работе [Keller, 1992, раздел 6.1] рассматривается нелинейная задача на собственные значения, связанная с исследованием процесса трения при наличии смазки. В качестве математической модели используется ОДУ первого порядка

$$\varepsilon y' = \sin^2(x) - \lambda \frac{\sin^4(x)}{y}, \tag{3.19}$$

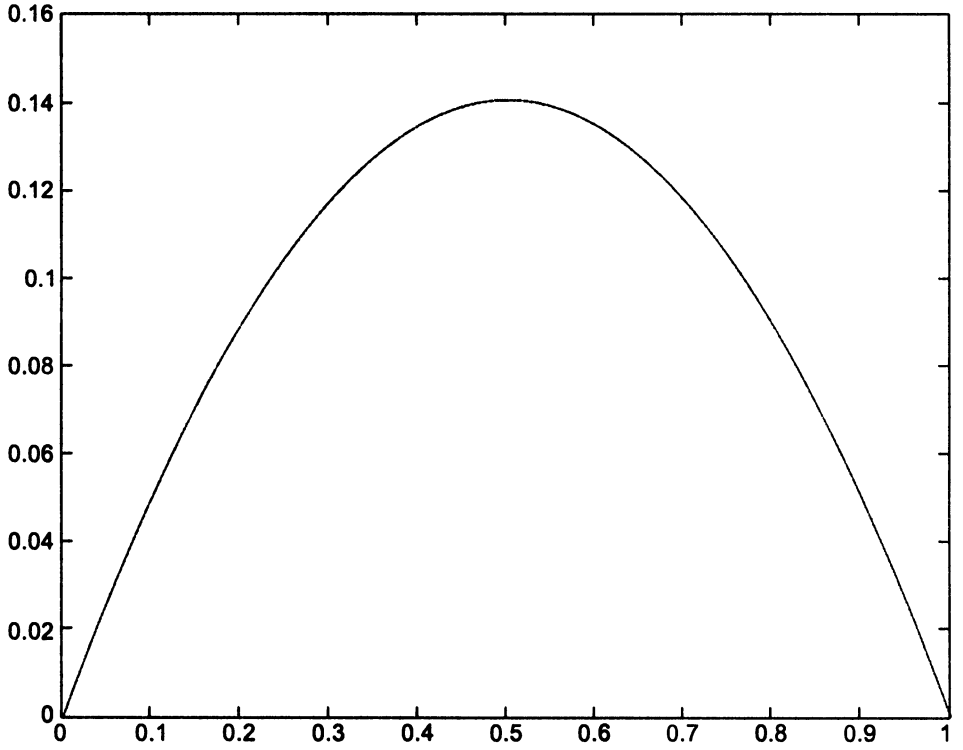

Рис. 3.3. График решения уравнения Брату, построенный по 100 равноразнесенным точкам.

где ε — известный параметр, а значение неизвестного параметра λ должно быть выбрано так, чтобы уравнение имело решение, удовлетворяющее двум граничным условиям

$$y\left(-\tfrac{\pi}{2}\right) = 1, \quad y\left(\tfrac{\pi}{2}\right) = 1. \tag{3.20}$$

Первое равенство может рассматриваться как граничное условие для исследуемого ОДУ первого порядка, а второе — в качестве условия, используемого для определения неизвестного параметра. Неизвестные параметры нередко используются при исследовании физических моделей или же они вводятся в затруднительных ситуациях для облегчения выполнения вычислительного процесса нахождения решений. Численная процедура `bvp4c` позволяет легко решить ЗГУ с неизвестными параметрами, и далее мы проиллюстрируем это на конкретных примерах. Большинство других процедур не предоставляют этой возможности, но, тем не менее, подобные задачи могут быть решены с их использованием, и мы покажем, как это можно сделать.

При численном решении ЗГУ с неизвестными параметрами наряду с оценками для решения необходимо задавать оценки для этих параметров. Эти оценки определяют критерии выбора множества допустимых значений параметров и облегчают их нахождение. Вектор, составленный из этих оценок, передается в процедуру `bvpinit` в виде третьего аргумента. Соответственно, вектор неизвестных

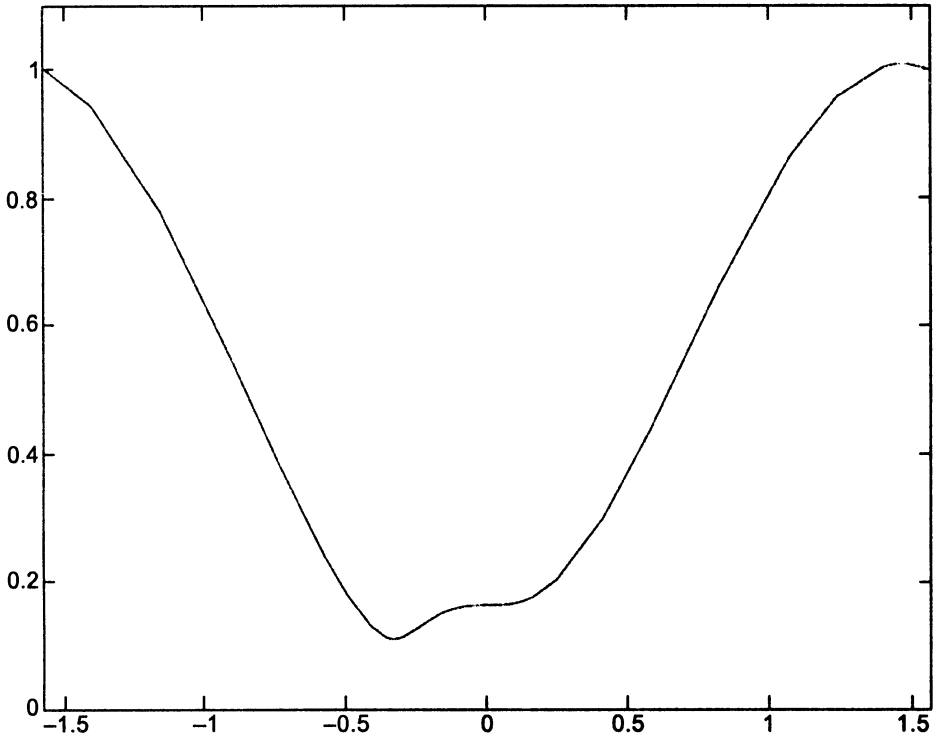

Рис. 3.4. Решение задачи о трении при наличии смазки, вычисленное при $\varepsilon = 0.1$.

параметров следует использовать в качестве третьего аргумента процедур, вычисляющих ОДУ и невязки по граничным условиям. Это необходимо делать даже в том случае, если в этих функциях значения неизвестных параметров не используются. При наличии аргументов, соответствующих неизвестным параметрам, в структуре-решении появляется дополнительное поле `parameters`, содержащее вектор параметров, вычисленный в численной процедуре. ЗГУ с ОДУ (3.19) и граничными условиями (3.20) решается с использованием программы `ch3ex2.m` при $\varepsilon = 0.1$. В качестве оценки параметра λ выступает 1, а в качестве оценки решения — постоянное значение $y(x) \approx 0.5$. Заметим, что `lambda` входит в список входных аргументов функции `bcs` несмотря на то, что это значение в ней не используется. Программа выводит на экран компьютера вычисленное значение параметра $\lambda \approx 1.01864$, а также график решения, изображенный на рисунке 3.4.

```
function sol = ch3ex2

solinit = bvpinit(linspace(-pi/2,pi/2,20),0.5,1);
sol = bvp4c(@ode,@bcs,solinit);
fprintf('lambda = %g.\n',sol.parameters)
plot(sol.x,sol.y(1,:));
axis([-pi/2 pi/2 0 1.1]);
```

```
%================================================
function dydx = ode(x,y,lambda)
epsilon = 0.1;
dydx = (sin(x)^2 - lambda*sin(x)^4/y)/epsilon;

function res = bcs(ya,yb,lambda)
res = [ ya-1; yb-1 ];
```

Большинство численных процедур не приспособлено для решения ЗГУ с неизвестными параметрами, однако эти задачи могут быть легко переформулированы так, что их решение с использованием подобных численных процедур станет возможным. В рассматриваемой здесь задаче мы имеем неизвестную функцию $y_1(x) = y(x)$ и неизвестный параметр λ. Введем дополнительную неизвестную функцию $y_2(x) = \lambda$ и добавим к исследуемому ОДУ дифференциальное уравнение для $y_2(x)$, решением которого является тривиальное решение $y_2(x) = \lambda$

$$y_1' = \varepsilon^{-1}\left[\sin^2(x) - y_2\frac{\sin^4(x)}{y_1}\right],$$
$$y_2' = 0.$$

Эта система ОДУ с граничными условиями (3.20) определяет новую ЗГУ без неизвестных параметров. При использовании многих численных процедур необходимо определять функции для аналитического вычисления частных производных правой части ОДУ и выражений для граничных условий по переменным состояния системы. В этом случае наряду с введением новой переменной для неизвестного параметра необходимо также определить функции для аналитического вычисления частных производных правой части ОДУ и выражений для граничных условий по этой новой переменной (т. е. частные производные по параметрам).

ПРИМЕР 3.5.3

В работе [Seydel, 1988] исследовался процесс распространения нервных импульсов, описывающийся системой ОДУ

$$y_1' = 3\left(y_1 + y_2 - \frac{y_1^3}{3} - 1.3\right),$$
$$y_2' = -\frac{1}{3}(y_1 - 0.7 + 0.8y_2)$$

с периодическими граничными условиями

$$y_1(0) = y_1(T),\ y_2(0) = y_2(T).$$

В условиях, когда период T нам известен, эти два неразделенных граничных условия совместны, если решение имеет период T. В упражнении 3.30 рассматривается именно такая ситуация. Однако если при решении задачи мы не задаем величину периода решения, наряду с периодическим решением периода T необходимо вычислить также и сам период T, выступающий в этой задаче в качестве неизвестного параметра. Для этого необходимо задать другое граничное условие. Следует

отметить, что если $(y_1(t), y_2(t))$ — периодическое решение приведенных выше автономных дифференциальных уравнений, то $(u_1(t), u_2(t)) = (y_1(t+\gamma), y_2(t+\gamma))$ также является решением этих уравнений при любой константе γ. В качестве искомого граничного условия можно выбрать равенство одной из компонент решения нулю, например, $y_1(0) = 0$. В этом случае периодическое граничное условие $y_1(0) = y_1(T)$ может быть заменено на $y_1(T) = 0$, поскольку это условие является эквивалентным и разделенным. Таким образом, задача состоит в решении ОДУ при граничных условиях

$$y_1(0) = 0,\ y_1(T) = 0,\ y_2(0) = y_2(T),$$

одно из которых является неразделенным. Численная процедура `bvp4c` допускает решение задач с неразделенными граничными условиями, однако большинство других численных процедур не предоставляют этой возможности. После решения этой задачи с использованием `bvp4c`, мы обсудим вопрос о преобразовании задачи с неразделенными граничными условиями к виду, допускающему использование подобных численных процедур. Основная трудность, с которой приходится сталкиваться при решении рассматриваемой задачи, заключается в том, что длина интервала $[0, T]$ неизвестна. В основанной на использовании метода пристрелки численной процедуре `D02SAF` [NAG, 2002] все неизвестные — включая конечные точки интервала, граничные значения и все параметры — рассматриваются как неизвестные параметры и поэтому указанная процедура может быть непосредственно использована при решении рассматриваемой задачи. В этом смысле `D02SAF` является единственной на сегодняшней день численной процедурой, предоставляющей эту возможность. Для использования других процедур (включая `bvp4c`) необходимо переформулировать исходную задачу, определив ее на фиксированном интервале. Если в качестве независимой переменной t использовать $x = t/T$, система ОДУ примет следующий вид

$$\frac{dy_1}{dx} = 3T\left(y_1 + y_2 - \frac{y_1^3}{3} - 1.3\right),$$
$$\frac{dy_2}{dx} = -\frac{T}{3}(y_1 - 0.7 + 0.8y_2).$$

Новая задача поставлена на интервале $[0, 1]$ с новыми граничными условиями

$$y_1(0) = 0,\ y_1(1) = 0,\ y_2(0) = y_2(1),$$

и она может быть легко решена с использованием программы `ch3ex3.m`.

```
function sol = ch3ex3

solinit = bvpinit(linspace(0,1,5),@guess,2*pi);
sol = bvp4c(@ode,@bc,solinit);
T = sol.parameters;
fprintf('The computed period T = %g.\n',T);
plot(T*sol.x,sol.y(1,:),T*solinit.x,solinit.y(1,:),'ro')
legend('Computed Solution','Guessed Solution');
```

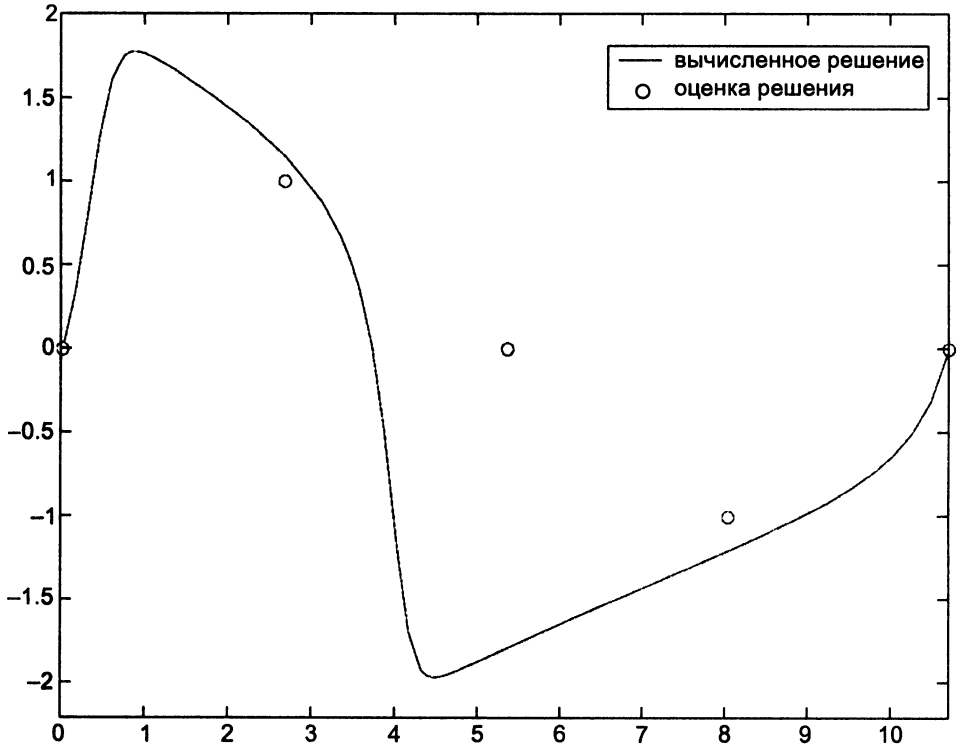

Рис. 3.5. Периодическое решение в модели нервного импульса.

```
axis([0 T -2.2 2]);

%=================================================
function v = guess(x)
v = [ sin(2*pi*x); cos(2*pi*x)];

function dydt = ode(x,y,T);
dydt = [ 3*T*(y(1) + y(2) - (y(1)^3)/3 - 1.3)
         -(T/3)*(y(1) - 0.7 + 0.8*y(2))      ];

function res = bc(ya,yb,T)
res = [ ya(1); yb(1); (ya(2) - yb(2)) ];
```

В качестве оценок используются $T = 2\pi$, $y_1(x) \approx \sin(2\pi x)$ и $y_2(x) \approx \cos(2\pi x)$. После вычисления решения независимая переменная x заново масштабируется в исходную независимую переменную $t = Tx$ и график решения выводится на экран компьютера (см. рис. 3.5). Заметим, что заданная начальная оценка для периода T оказалась неточной: вычисленное программой значение $T \approx 10.7106$. Рассматриваемая задача очень проста и для обеспечения сходимости численной процедуры более точные оценки не потребовались.

Далее граничное условие $y_2(0) = y_2(T)$ будет использовано при решении проблемы, связанной с наличием неразделенных граничных условий, когда существующие численные процедуры решения ЗГУ не могут быть применены. Идея заключается во введении новой переменной $y_3(t) \equiv y_2(T)$ и добавлении в существующую систему ОДУ соответствующего дифференциального уравнения $y_3' = 0$. Поскольку $y_3(t)$ сохраняет постоянное значение, граничное условие $y_2(0) = y_2(T)$ эквивалентно граничному условию $y_2(0) = y_3(0)$ и мы получаем другое граничное условие $y_2(T) = y_3(T)$. Таким образом, неразделенное граничное условие заменено на два разделенных граничных условия, введена одна новая переменная и одно новое ОДУ. Окончательно получаем систему ОДУ

$$\begin{aligned}
\frac{dy_1}{dx} &= 3\left(y_1 + y_2 - \frac{y_1^3}{3} - 1.3\right), \\
\frac{dy_2}{dx} &= -\frac{1}{3}(y_1 - 0.7 + 0.8y_2), \\
\frac{dy_3}{dx} &= 0
\end{aligned}$$

с (разделенными) граничными условиями

$$y_1(0) = 0,\ y_1(T) = 0,\ y_2(0) = y_3(0),\ y_2(T) = y_3(T).$$

В общем случае наличие одного неразделенного граничного условия приводит к добавлению одной компоненты решения, одного дополнительного граничного условия и одного тривиального ОДУ. В упражнении 3.30 рассматривается другой аналогичный пример.

ПРИМЕР 3.5.4

В этом примере, взятом из работы [Gladwell, 1979a] (с исправлением несущественной ошибки), показывается, как при решении ЗНУ с сингулярностями могут возникать неизвестные параметры, а также как могут быть решены задачи, поставленные на бесконечном интервале. Для решения рассматриваемого дифференциального уравнения в частных производных (ДУЧП), описывающего движение потока жидкости, применяется метод подобия. Представленный ниже анализ может показаться излишне подробным, однако в случае, когда вам необходимо решить аналогичную задачу, его детали могут оказаться для вас полезными. Система ОДУ

$$3yy'' = 2(y' - z),\ z'' = -yz' \tag{3.21}$$

рассматривается с граничными условиями

$$y(0) = 0,\ z(0) = 1 \tag{3.22}$$

и

$$y'(+\infty) = 0,\ z'(+\infty) = 0. \tag{3.23}$$

Первое уравнение сингулярно в начале координат, т. к. $y(0) = 0$. Для разрешения этой проблемы найдем с требуемой точностью аналитические аппроксимации компонент решения $y(x)$ и $z(x)$ на интервале $[0, \delta]$, где $\delta > 0$ — некоторая малая

константа. С учетом того, что мы имеем два уравнения второго порядка и только два граничных условия в начале координат, в выражениях для этих аппроксимаций будут присутствовать два свободных параметра, в качестве которых могут выступать неизвестные граничные условия $y'(0)$ и $z'(0)$. Далее, на интервале $[\delta, \infty)$, выполняется численное решение ЗГУ, определяемой теми же ОДУ (3.21) с теми же граничными условиями на бесконечности (3.23). Заметим, что на этом интервале уравнения не являются сингулярными. Граничные условия в начале координат (3.22) заменяются в этой новой задаче на условия равенства значений численного решения, вычисленных в δ, значениям, вычисленным с использованием аналитических аппроксимаций в той же точке. Свободные параметры в аналитических аппроксимациях выступают в роли неизвестных параметров ЗГУ. Значения этих параметров определяются в результате численного решения этой задачи. С учетом всего вышесказанного можно найти численные значения $y(x)$ и $z(x)$ на всем интервале $[0, \infty)$.

Если рассматриваемое дифференциальное уравнение нелинейно, поведение его решений вблизи сингулярной точки предсказать непросто. В общем случае при решении этой проблемы можно руководствоваться физическими аспектами рассматриваемой задачи, но можно также воспользоваться аналитической аппроксимацией решения рядом Тейлора. Это можно легко сделать с помощью компьютерных средств символьных вычислений. Полученный таким образом результат позволяет получить некоторую полезную информацию о поведении решения. В нижеследующей программе, в которой используется имеющийся в MATLAB пакет символьных вычислений (MATLAB Symbolic Toolbox), разложение решения в ряд Тейлора с несколькими членами подставляется в уравнение (3.21).

```
syms x y z A B C D E F eqn1 eqn2
y = 0 + A*x + B*x^2 + C*x^3;
z = 1 + D*x + E*x^2 + F*x^3;
eqn1 = collect(3*y*diff(y,2,'x') - 2*(diff(y,'x') - z))
eqn2 = collect(diff(z,2,'x') + y*diff(z,'x'))
```

Заметим, что при вычислениях уже учтены граничные условия (3.22). Результаты работы этой программы (слегка отредактированные) представлены ниже

```
eqn1 = 18*C^2*x^4+(24*B*C+2*F)*x^3
       +(-6*C+18*A*C+6*B^2+2*E)*x^2+(-4*B+6*A*B+2*D)*x-2*A+2
eqn2 = 3*C*F*x^5+(3*B*F+2*C*E)*x^4+(3*A*F+2*B*E+C*D)*x^3
       +(2*A*E+B*D)*x^2+(6*F+A*D)*x+2*E
```

Поскольку нас интересует поведение решения при $x \to 0$, чем больше степень x, тем меньше влияние соответствующего члена на поведение решения. Наша цель заключается в том, чтобы разложение для решения как можно более точно удовлетворяло уравнениям, и поэтому нам необходимо выбрать коэффициенты этого разложения так, чтобы как можно больше последовательных членов в этом разложении были равны нулю (начиная с члена наименьшего порядка). Из разложения видно, что для исключения постоянных членов необходимо выбрать $A = 1$ и $E = 0$. Далее, для исключения членов при x следует положить $D = -B$ и $F = -D/6$.

Члены при x^2 не могут быть исключены без исключения других членов в разложении Тейлора. Таким образом, мы можем заключить, что при малых значениях x мы имеем $y(x) = x + Bx^2 + \ldots$ и $z(x) = 1 - Bx + (B/6)x^3 + \ldots$. В этих разложениях коэффициент B выступает в роли свободного параметра.

К сожалению, полученные выше разложения не могут служить в качестве соответствующих аппроксимаций, т. к. необходимые нам аппроксимации должны содержать два свободных параметра, а в полученных выше имеется лишь один. Возможно, второй свободный параметр является коэффициентом при каком-либо члене более высокого порядка в разложении Тейлора, но вероятнее всего, мы совершили ошибку, предположив, что решение рассматриваемой задачи может быть разложено в ряд Тейлора. Попытаемся разложить решение по степеням x, в предположении, что показатели этих степеней являются дробными величинами. Эта задача более трудная, и мы попытаемся решить ее с использованием ряда с небольшим числом членов в надежде, что этот подход быстро увенчается успехом. Итак, пусть решение имеет вид

$$y(x) = ax^{\alpha} + bx^{\beta} + \ldots,$$
$$z(x) = 1 + cx^{\gamma} + dx^{\delta} + \ldots.$$

С учетом предположений $0 < \alpha < \beta$ и $0 < \gamma < \delta$, эти разложения удовлетворяют граничным условиям (3.22). Подставляя эти выражения в первое дифференциальное уравнение (3.21), получаем равенство

$$0 = 3[ax^{\alpha} + bx^{\beta} + \ldots][\alpha(\alpha-1)ax^{\alpha-2} + \beta(\beta-1)bx^{\beta-2} + \ldots] - \\ -2[(\alpha a x^{\alpha-1} + \beta b x^{\beta-1} + \ldots) - (1 + cx^{\gamma} + dx^{\delta} + \ldots)].$$

Если $\alpha < 1$, членом наименьшего порядка является $3\alpha(\alpha-1)a^2x^{2\alpha-2}$. Он не может быть исключен, поскольку мы предположили, что равенства $a = 0$ и $\alpha = 0$ не могут быть выполнены. С другой стороны, если предположить, что $\alpha = 1$, мы имеем два члена наименьшего порядка $-2[(ax^0) - (1)]$, которые могут быть исключены при $a = 1$. Подставив решение во второе уравнение, получаем

$$0 = \gamma(\gamma-1)cx^{\gamma-2} + \delta(\delta-1)dx^{\delta-2} + \ldots + \\ +(x + bx^{\beta} + \ldots)(\gamma c x^{\gamma-1} + \delta d x^{\delta-1} + \ldots).$$

В этом случае членом наименьшего порядка является $\gamma(\gamma-1)cx^{\gamma-2}$. Для его исключения следует положить $\gamma = 1$. Тогда вышеприведенное равенство упрощается

$$0 = \delta(\delta-1)dx^{\delta-2} + \ldots + cx + \ldots.$$

Для исключения члена при $x^{\delta-2}$ коэффициент δ должен быть выбран так, чтобы полученное значение степени было равно степени «следующего» члена. Легко видеть, что при $\delta = 3$ эти два члена наименьшего порядка могут быть исключены путем выбора $d = -c/6$, где коэффициент c выступает в качестве свободного параметра. Вернемся к рассмотрению результата подстановки разложения решения в первое уравнение и используем всю новую информацию. Подставив найденные коэффициенты, получаем

$$3[x + bx^{\beta} + \ldots][\beta(\beta-1)bx^{\beta-2} + \ldots] = 2\left[(\beta b x^{\beta-1} + \ldots) - \left(cx - \tfrac{c}{6}x^{\delta} + \ldots\right)\right].$$

Если $\beta < 2$, наименьшей степенью является $\beta - 1$. Для исключения соответствующих членов следует приравнять коэффициенты при $x^{\beta-1}$

$$3b\beta(\beta - 1) = 2b\beta.$$

Из этого уравнения находим $\beta = \frac{5}{3}$. Оставшийся коэффициент b рассматриваем в качестве свободного параметра.

Наиболее трудоемкий этап нахождения разложений решения позади. Можно заметить, что полученное разложение решения отличается от представленного выше разложения в ряд Тейлора лишь членом $bx^{5/3}$ в выражении для $y(x)$. Коэффициент b представляет собой второй свободный параметр (в дополнение к B), которого нам недоставало при использовании подхода, использующего разложение в ряд Тейлора. Важным моментом анализа было осознание того факта, что искомое разложение решения должно содержать члены, являющиеся степенями $x^{1/3}$. С учетом этого результата, можно легко определить дополнительные члены. Например, для вычисления следующих двух членов можно воспользоваться следующей программой

```
syms x y z b c d e eqn1 eqn2
y = x + b*x^(5/3) + d*x^2;
z = 1 + c*x - (c/6)*x^3 + e*x^(10/3);
eqn1 = collect(3*y*diff(y,2,'x') - 2*(diff(y,'x') - z))
eqn2 = collect(diff(z,2,'x') + y*diff(z,'x'))
```

В отредактированном выводе этой программы оставлены только члены наименьшего порядка

```
eqn1 = ... +10/3*b^2*x^(4/3)+(2*d+2*c)*x
eqn2 = ... +b*c*x^(5/3)+70/9*e*x^(4/3)
```

Для исключения членов наименьшего порядка положим $d = -c$ и $e = 0$. Окончательно имеем

$$y(x) = x + bx^{5/3} - cx^2 - \frac{5b^2}{7}x^{7/3} + \frac{7bc}{6}x^{8/3} + \left(\frac{95b^3}{126} - \frac{c^2}{2}\right)x^3 + \ldots,$$

$$z(x) = 1 + cx - \frac{c}{6}x^3 - \frac{9bc}{88}x^{11/3} + \frac{c^2}{12}x^4 + \frac{9b^2c}{182}x^{13/3} - \frac{3bc^2}{44}x^{14/3} + \ldots.$$

Граничные условия на бесконечности могут быть аппроксимированы равенствами $y'(\Delta) = 0$ и $z'(\Delta) = 0$, где Δ — некоторая большая константа. Подобный подход часто бывает успешным, но его использование, например, в программе `ch3ex4.m` не позволяет получить решение задачи, т. к. вычислительный процесс оказывается неустойчивым. Очевидно, в численную процедуру необходимо ввести некоторую дополнительную информацию о поведении решения вблизи бесконечности. Для этого прежде всего заметим, что существует постоянное решение исследуемой системы ОДУ, удовлетворяющее граничным условиям на бесконечности: $y(x) \equiv \alpha$ и $z(x) \equiv 0$, где α — произвольная константа. (Решение подобного вида с $\alpha = \delta \approx y(\delta)$ используется в качестве оценки в программе `ch3ex4.m`.) Для исследования поведения решений ОДУ, близких к этому решению, линеаризуем уравнения

в окрестности $(\alpha, 0)$, или, эквивалентно, найдем решение следующего вида

$$y(x) \sim \alpha + \beta e^{\gamma x},\ z(x) \sim 0 + \delta e^{\varepsilon x}$$

при $x \to \infty$. Для $y(x)$ подобного вида справедливо равенство

$$y'(x) \sim \gamma\beta e^{\gamma x}.$$

Для того, чтобы получить нетривиальное решение, удовлетворяющее граничному условию $y'(\infty) = 0$, необходимо $\gamma < 0$. Аналогично, из другого граничного условия (на бесконечности) получаем, что необходимо $\varepsilon < 0$. Подставим решение указанного вида во второе дифференциальное уравнение

$$\varepsilon^2\delta e^{\varepsilon x} \sim -(\alpha + \beta e^{\gamma x})\varepsilon\delta e^{\varepsilon x}.$$

После сокращения общих множителей, получаем

$$\varepsilon \sim -(\alpha + \beta e^{\gamma x}) \sim -\alpha.$$

Поскольку $\varepsilon < 0$, необходимо $\alpha > 0$. Далее, подставим решение указанного вида в первое дифференциальное уравнение

$$3(\alpha + \beta e^{\gamma x})\gamma^2\beta e^{\gamma x} \sim 2(\gamma\beta e^{\gamma x} - \delta e^{-\alpha x}).$$

Поделив правую и левую части этого уравнение на $e^{\gamma x}$, получаем

$$3(\alpha + \beta e^{\gamma x})\gamma^2\beta \sim 2(\gamma\beta - \delta e^{-(\alpha+\gamma)x}) \tag{3.24}$$

и, рассматривая поведение решения при $x \to \infty$, приходим к выводу, что необходимо $(\alpha + \gamma) \geqslant 0$. Если $(\alpha + \gamma) > 0$, то переходя в (3.24) к пределу, получаем

$$3\alpha\gamma^2\beta = 2\gamma\beta$$

и, следовательно, $3\alpha\gamma = 2$. Однако это равенство не может быть выполнено, поскольку $\alpha > 0$ и $\gamma < 0$. Таким образом, мы приходим к выводу, что $(\alpha + \gamma) = 0$, т. е. $\gamma = -\alpha$. С учетом этого предположения снова перейдем в (3.24) к пределу, в результате чего получаем равенство

$$3\alpha\gamma^2\beta = 2(\gamma\beta - \delta),$$

которое может быть выполнено при

$$\delta = -\alpha\beta - \tfrac{3}{2}\alpha^3\beta.$$

Итак, мы нашли асимптотическое решение рассматриваемой системы ОДУ с двумя свободными параметрами α и β, удовлетворяющее граничным условиям на бесконечности

$$y(x) \sim \alpha + \beta e^{-\alpha x},\ z(x) \sim -\left(\alpha\beta + \tfrac{3}{2}\alpha^3\beta\right) e^{-\alpha x}.$$

В программе `ch3ex4.m` в качестве аппроксимаций компонент решения $y(x)$ и $z(x)$ при малых x мы использовали степенные ряды, включающие члены до порядка $O(x^{8/3})$ и $O(x^{11/3})$, соответственно. В этом случае оба старших члена разложения дифференциальных уравнений, фактически являющиеся невязками, имеют порядок $O(x^2)$. Граничные условия в начале координат заменены на требование равенства значений численного решения соответствующим значениям аппроксимирующих рядов в точке $x = \delta$. Коэффициенты b и c, присутствующие в выражениях для рядов, передаются в вычислительную процедуру как компоненты вектора `P`. Представление рассматриваемой системы в виде системы уравнений первого порядка зависит от четырех неизвестных. Остальные неизвестные, производные $y'(x)$ и $z'(x)$, аппроксимируются производными рядов для $y(x)$ и $z(x)$. Вектор аппроксимаций неизвестных соответствующими рядами вычисляется в функции `series`. Эта функция используется не только при вычислении невязок по граничным условиям, но и при вычислении решения в любой точке из промежутка $[0, \delta]$. Асимптотическое решение с неизвестными параметрами может быть использовано при некоторой большой Δ, аналогично тому, как используются аппроксимации решения рядами при малой δ в случае наличия сингулярности в начале координат. Однако в целях иллюстрации методики применения численных процедур, неприспособленных для решения задач с неизвестными параметрами, заметим, что асимптотические решения удовлетворяют равенствам

$$y(x) \sim \alpha, \quad z'(x) \sim -\alpha z(x),$$

из которых получаем граничное условие

$$z'(x) \sim -y(x)z(x),$$

независящее от неизвестных параметров. Поведение старшего члена в разложении компоненты решения $y(x)$ на бесконечности определяется граничным условием $y'(\infty) = 0$, которое также не зависит от неизвестных параметров. Необходимо иметь ввиду, что в общем случае при решении задачи может возникнуть необходимость в предоставлении дополнительной информации о поведении решения на бесконечности. Однако в рассматриваемой программе `ch3ex4.m` мы ограничились наложением граничных условий

$$y'(\Delta) = 0, \; z'(\Delta) = -y(\Delta)z(\Delta),$$

что оказалось достаточным при умеренных значениях Δ. В упражнении 3.9 рассматривается аналогичная задача с сингулярностями в начале координат и на бесконечности.

Численное решение ЗГУ на промежутке $[\delta, \Delta]$ может быть выполнено непосредственно. Однако при этом всегда полезно выполнить несколько экспериментов со значениями конечных точек указанного промежутка, чтобы убедиться в правильности их выбора. Было бы естественным выбрать начальную точку вблизи сингулярной точки, а в качестве конечной — некоторую большую величину, но это может привести к возникновению проблем, которых мы пытались избежать, выполняя представленный выше анализ: различные решения ОДУ могут оказаться

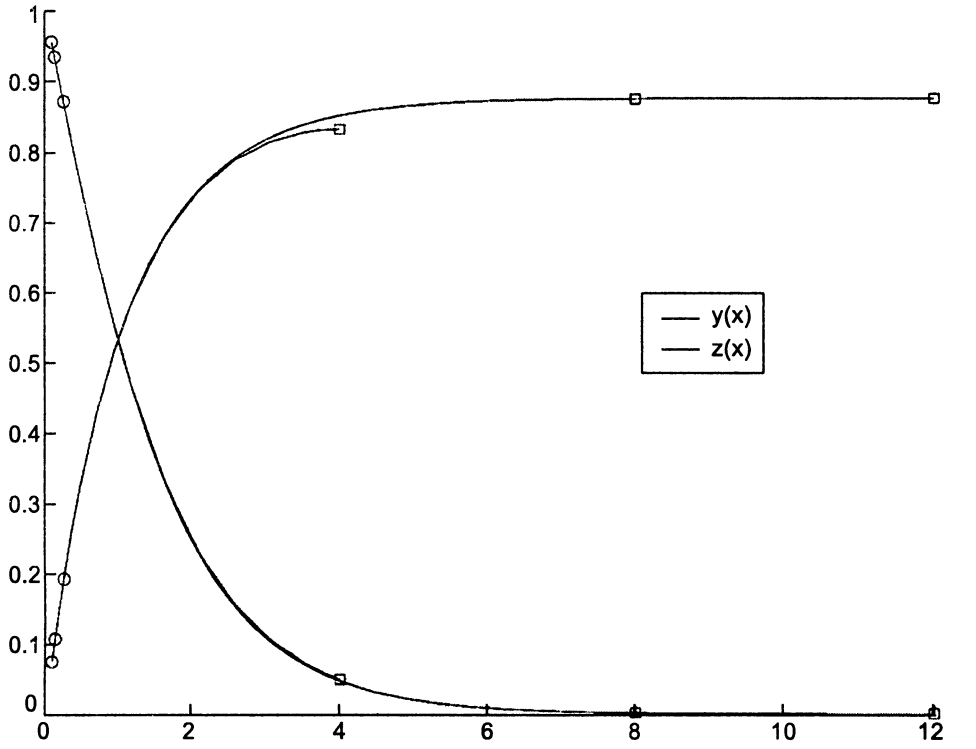

Рис. 3.6. Решение ЗГУ, определенной на бесконечном интервале и имеющей сингулярность в начале координат.

численно неразличимыми в процессе работы численной процедуры. В программе `ch3ex4.m` рассматриваемая ЗГУ решается на трех интервалах, для каждого из которых выводится график полученного решения для последующего анализа пользователем. В подобных обстоятельствах оказываются очень полезными те возможности, которые заложены в функции `bvpinit`. После решения задачи на одном интервале полученное решение можно рассматривать как хорошую оценку, используемую при решении задачи на несколько большем интервале. В этом случае структура-решение и этот новый интервал используются в качестве новых аргументов функции `bvpinit`, формирующей структуру-оценку, используемую для решения задачи на новом интервале. Таким образом, вычисляемое решение автоматически расширяется на больший интервал путем экстраполяции. Длина интервала, на который может быть продолжено решение с использованием этого подхода, зависит от скорости изменения самого решения и обусловлена также хорошо известными проблемами, связанными с полиномиальной экстраполяцией. Целостность вычисленных решений (см. рис. 3.6) показывает, что граничные точки, полученные по представленному алгоритму, расположены настолько близко к сингулярным точкам, что наши аппроксимации граничных условий можно считать приемлемыми.

```
function ch3ex4
% y(1) = y(x), y(2) = z(x), y(3) = y'(x), y(4) = z'(x)
global d
hold on
for i = 1:3
    D = 4*i; d = 1/D;
    if i == 1
        P = [-1; 0.5]; % Guess parameters P = [b; c]:
        solinit = bvpinit(linspace(d,D,5),[d; 0; 0; 0],P);
    else
        solinit = bvpinit(sol,[d,D]);
    end
    sol = bvp4c(@odes,@bcs,solinit);
    plot(sol.x,sol.y(1:2,:),sol.x(end),sol.y(1:2,end),'ro',...
         sol.x(1),sol.y(1:2,1),'ko');
    legend('y(x)','z(x)',0);
    axis([0 12 0 1]);
    drawnow
end
hold off

%============================================================
function dydx = odes(x,y,P)
dydx = [ y(3); y(4); 2*(y(3) - y(2))/(3*y(1)); -y(1)*y(4) ];

function res = bcs(ya,yb,P)
global d
res = zeros(6,1);
res(1:4) = ya - series(d,P);
res(5:6) = [ yb(3); (yb(4) - (- yb(1)*yb(2))) ];

function y = series(x,P)
b = P(1); c = P(2);
yx  = x + b*x^(5/3) - c*x^2 - (5/7)*b^2*x^(7/3) ...
      + (7/6)*b*c*x^(8/3);
ypx = 1 + (5/3)*b*x^(2/3) - 2*c*x - (5/3)*b^2*x^(4/3)...
      + (28/9)*b*c*x^(5/3);
zx  = 1 + c*x - (1/6)*c*x^3 - (9/88)*b*c*x^(11/3);
zpx = c - (1/2)*c*x^2 - (3/8)*b*c*x^(8/3);
y = [ yx; zx; ypx; zpx];
```

Очень часто наиболее трудным этапом численного решения ЗГУ является нахождение начальной оценки решения, при которой вычислительный процесс сходится. Выше мы рассмотрели пример решения ЗГУ на нескольких интервалах, как способ, позволяющий выявить характерные свойства решения. Этот пример может рассматриваться также в контексте метода *продолжения* решения. В основе этого метода лежит фундаментальный результат из теории дифференциальных уравнений, который нестрого говоря заключается в том, что решение некоторой ЗГУ может служить в качестве хорошей оценки для решения другой ЗГУ,

отличающейся от первой лишь небольшим отклонением значений параметров рассматриваемой задачи. Программа `ch3ex4.m` является иллюстрацией продолжения решения на больший интервал. Если вы испытываете затруднения при нахождении оценки решения, при которой обеспечивается сходимость вычислительного процесса, вероятнее всего эта задача может быть решена на интервале меньшей длины. В этом случае следует попытаться последовательно решить несколько ЗГУ, для каждой из которых (разумеется, начиная со второй) решение предыдущей задачи на меньшем интервале используется в качестве оценки решения на новом, расширенном интервале. Если эта процедура увенчается успехом, вы наконец придете к задаче, определенной на интересующем вас интервале, имея достаточно хорошую оценку решения, при которой обеспечивается сходимость вычислительного процесса. Разумеется, этот подход невозможно использовать для того, чтобы продолжить интервал до бесконечности; сколь бы точна ни была ваша оценка, в какой-то момент численная процедура не сможет численно различить два различных решения. Длина интервала интегрирования служит в качестве параметра, по которому осуществляется процедура продолжения решения и полученные таким образом соответствующие численные решения, определенные на близких интервалах, будут также близки, но в общем случае это утверждение неверно. В частности, использование метода продолжения, в котором длина интервала рассматривается в качестве подобного параметра, может не увенчаться успехом, т. к. иногда малое изменение интервала интегрирования может привести к тому, что первоначальная корректно поставленная ЗГУ с хорошо определенным решением становится задачей, не имеющей решений или имеющей несколько решений.

ПРИМЕР 3.5.5

Задача Фишера, рассмотренная в подпараграфе 3.3.2, может быть использована в качестве иллюстрации ситуации, когда граничные условия заданы на бесконечности. Рассматриваемое ОДУ имеет вид

$$U'' + cU' + U(1 - U) = 0$$

и искомое решение должно удовлетворять граничным условиям

$$U(-\infty) = 1, \quad U(\infty) = 0.$$

Было установлено, что при указанных граничных условиях решение этой задачи не единственно. В результате анализа были определены граничные условия, зависящие от некоторого «большого» числа Z, при определенном выборе которого существует единственное решение

$$\frac{U'(-Z)}{U(-Z) - 1} - \alpha = 0 \quad \text{при} \quad \alpha = \frac{-c + \sqrt{c^2 + 4}}{2},$$

$$\frac{U(Z)}{e^{\beta Z}} - 1 = 0 \quad \text{при} \quad \beta = \frac{-c + \sqrt{c^2 - 4}}{2}.$$

В программе `ch3ex5.m` для величин Z, α и β используются переменные `infty`, `alpha` и `beta` соответственно. Граничные условия задаются непосредственно для системы первого порядка с двумя неизвестными $y_1(z) = U(z)$ и $y_2(z) = U'(z)$. Если

заданы адекватные граничные условия, численное решение этой ЗГУ не представляет проблем, но мы рассматриваем здесь эту задачу для того, чтобы проиллюстрировать ситуации, когда с использованием соответствующей численной процедуры найти решение оказывается невозможным. Мы также решим поставленную ЗГУ при различных значениях скорости волны c и предложим различные способы ускорения вычислительного процесса.

Точное измерение времени исполнения программы в MATLAB представляет собой непростую задачу и полученные результаты существенным образом зависят от используемого компьютерного оборудования и его конфигурации. В этой связи мы хотели бы выработать некоторые общие принципы, позволяющие выявить влияние различных опций на полученный результат. Значения времени выполнения программ были получены ранее на однопользовательском компьютере при помощи команд вида

```
>> tic, ch3ex5; toc
```

При заданных по умолчанию значениях опций `options=[]` время выполнения программы `ch3ex5.m`, листинг которой представлен в конце обсуждения этого примера, оказалось равным 5.76 секунд.

Так же как и в других численных процедурах MATLAB, предназначенных для решения жестких ЗНУ, в `bvp4c` вычисляются матрицы Якоби с использованием конечных разностей. Часто вычисление функции $f(x,y)$ в правой части исследуемого ОДУ может быть легко векторизуемо по переменной x и массиву векторов y. Это позволяет существенно уменьшить время выполнения программы и поэтому численные процедуры решения жестких ЗНУ имеют опцию, соответствующую именно такому выбору пользователя. Естественно предположить, что использование этой опции при вызове численных процедур решения ЗГУ также позволит снизить время вычислений, но дополнительный выигрыш возможен, если функция в правой части соответствующего ОДУ также векторизована и по переменной x. Это обстоятельство обусловлено тем, что при дискретизации ОДУ в контексте решения ЗГУ все значения аргументов заранее известны. При решении ЗНУ пользователь имеет возможность написать процедуры вычисления функции $f(x,y)$ так, чтобы при заданном векторе $x = [x_1, x_2, \ldots]$ и соответствующем массиве векторов-столбцов $y = [y_1, y_2, \ldots]$, эта процедура возвращала массив векторов-столбцов $[f(x_1,y_1), f(x_2,y_2), \ldots]$. Выполненная подобным образом векторизация вычислений функции $f(x,y)$ часто позволяет существенно сократить время выполнения программы. Если в программе `ch3ex5.m` удалить символ комментария в строке, активирующей опцию `Vectorized`, время выполнения программы сокращается до 3.40 секунд. В относительных величинах этот результат представляется существенным и показывает, что выполнение векторизации вычислений может быть полезным. Разумеется, в рассматриваемом случае абсолютное значение времени выполнения программы мало и предпринимать какие-либо специальные усилия для его уменьшения не требуется. Тем не менее, процедура векторизации может быть выполнена без каких-либо проблем. Вычисление ОДУ сводится к скалярному выражению

```
dydz = [ y(2); -(c*y(2) + y(1)*(1 - y(1))) ];
```

т. е. векторизация выполняется путем замены векторов на массивы и операции умножения на операцию умножения для массивов (см. листинг программы `ch3ex5.m`).

При использовании большинства численных процедур решения ЗГУ необходимо писать отдельные процедуры для аналитического вычисления частных производных. Это неудобно для пользователя и в общем случае не является необходимым при решении стандартных задач с использованием численных процедур MATLAB и, в частности, при использовании численной процедуры `bvp4c`. Тем не менее, существует множество причин, по которым следует воспользоваться преимуществами аналитического вычисления частных производных. В частности, эти преимущества проявляются в следующей ситуации: система линейных ОДУ может быть записана в форме

$$y'(x) = J(x)y(x) + q(x)$$

и программная реализация соответствующей процедуры для вычисления матрицы Якоби непосредственно следует из этой формы представления, т. к. этой матрицей является $J(x)$. Другая причина заключается в том, что в общем случае при использовании аналитического вычисления частных производных численная процедура решения ЗГУ работает быстрее. Несмотря на высокую вычислительную эффективность процедуры `numjac`, используемой в `bvp4c` для численной аппроксимации частных производных, аналитическое вычисление частных производных позволяет повысить робастность численной процедуры решения сложных ЗГУ. Более того, решение некоторых особенно сложных задач оказывается возможным лишь при использовании аналитического представления частных производных. Программная реализация `bvp4c` допускает возможность задания подобного представления частных производных для ОДУ и/или граничных условий. В большинстве случаев более важным представляется задание частных производных для функции в правой части ОДУ, чем для граничных условий. Указатель на процедуру вычисления $\frac{\partial f}{\partial y}$ передается в численную процедуру решения ЗГУ в виде значения опции `FJacobian`. Аналогично, указатель на процедуру аналитического вычисления частных производных для граничных условий передается в численную процедуру решения ЗГУ в виде значения опции `BCJacobian`. Процедура для вычисления невязок по граничным условиям $g(ya, yb)$ имеет два векторных аргумента: аппроксимации значений решения на концах интервала ya и yb. Соответственно, процедура для вычисления частных производных для граничных условий (которую должен написать пользователь) должна возвращать две матричные величины: $\frac{\partial g}{\partial ya}$ и $\frac{\partial g}{\partial yb}$.

При написании процедуры вычисления матрицы Якоби

$$J = \frac{\partial f}{\partial y} = \left(\frac{\partial f_i}{\partial y_j}\right)$$

могут оказаться полезными нижеследующие комментарии. Предположим, что рассматривается система из d уравнений. Если матрица J является разреженной, ее следует инициализировать нулями как разреженную матрицу `J = sparse(d, d)`. После этого для каждого значения $i = 1, 2, \ldots, d$ следует проверить функцию f_i на предмет ее зависимости от компонент решения y_j. Для каждой компоненты y_j, от которой зависит функция f_i, следует выписать соответствующее аналитическое

выражение для частной производной и написать программный код для вычисления соответствующего элемента матрицы Якоби $J(i,j)$. Разумеется, компоненты матрицы J могут вычисляться в программе в любом удобном для вас порядке. Предлагаемый подход следует применять даже в случае, когда вы рассматриваете матрицу J как плотную. В этом случае единственное отличие заключается в том, что ее инициализация нулями должна выполняться с использованием команды `J = zeros (d, d)` (или просто `J = zeros(d)`). Функция для вычисления $f(x,y)$ должна возвращать вектор-столбец f. В случае, когда матрица Якоби не рассматривается как разреженная, соответствующие частные производные удобно вычислять по столбцам, т. е. последовательно для каждого значения $j = 1, 2, \ldots, d$

$$J(:,j) = \frac{\partial f}{\partial y_j} = \left(\frac{\partial f_i}{\partial y_j}\right).$$

После того, как вы получили выражения для всех частных производных, вы можете вычислить их значения в удобном для вас порядке. В пакете символьных вычислений MATLAB имеется процедура `jacobian`, которая может оказаться очень полезной при получении выражений для частных производных сложных функций. Использование этих средств мы проиллюстрируем на примере вывода выражений частных производных для граничных условий рассматриваемой ЗГУ. Для переменных, в которых будут храниться аналитические выражения для частных производных, удобно использовать имена вида `dBcdya`, соответствующие вычисляемой частной производной. Нижеследующая программа написана с учетом этого соглашения о именах переменных

```
syms res ya1 ya2 yb1 yb2 alpha beta infty
res = [ ya2/(ya1 - 1) - alpha
        yb1/exp(beta*infty) - 1 ];
dBCdya = jacobian(res,[ya1; ya2])
dBCdyb = jacobian(res,[yb1; yb2])
```

Результатом работы этой программы является следующий листинг

```
dBCdya = [ -ya2/(ya1-1)^2,    1/(ya1-1)]
         [ 0              ,           0]

dBCdyb = [ 0                ,         0]
         [ 1/exp(beta*infty),         0]
```

После незначительного редактирования этот листинг может быть преобразован в текст процедуры `bcJac`, используемой в программе `ch3ex5.m`.

Для решения рассматриваемой ЗГУ с использованием программы `ch3ex5.m` при значениях опций, заданных по умолчанию, за исключением опции `FJacobian`, потребовалось 4.72 секунды. Напомним, что при численном вычислении якобиана, т. е. без использования опции `FJacobian`, для решения этой задачи потребовалось 5.76 секунд. При опции `BcJacobian` для решения ЗГУ потребовалось 4.56 секунд В некоторых случаях к более существенному уменьшению времени выполнения программы приводит не аналитическое вычисление матриц Якоби, а векторизация вычислений. В других случаях эффект от использования векторизации может

оказаться ничтожно малым по сравнению с эффектом от использования аналитического представления частных производных. Эти опции независимы и вы можете пользоваться ими по своему усмотрению. Более того, вы можете одновременно использовать обе эти опции. Например, в рассматриваемом случае выполнение векторизации вычисления функции в правой части ОДУ и использование аналитических выражений для частных производных позволило уменьшить время выполнения программы до 2.31 секунд. Таким образом, можно заключить, что если это оказывается возможным, то в распоряжение численной процедуры следует предоставить максимум информации, чтобы уменьшить время вычислений. Более того, эти дополнительные усилия могут быть вознаграждены впоследствии. Например, в рассматриваемом примере, если вы планируете решить задачу при нескольких значениях параметров, полезно выполнить эту работу, несмотря на малость времени вычисления решения для одного значения этого параметра.

После того, как мы научились решать нашу задачу достаточно эффективно, выполним несколько численных экспериментов. Заметим, что в программе `ch3ex5.m` задается значение `infty`, зависящее от скорости волны c. Может показаться, что если в качестве этой величины использовать достаточно большое значение, то рассматриваемая задача будет фактически решена для всех интересующих нас значений скорости волны c. Однако если, например, положить `infty=250`, то процесс вычислений прерывается с выводом сообщения об ошибке[1]

```
?? Error using ==> bvp4c
Unable to refine the mesh any further --
the Jacobian of the collocation equations is singular
```

При меньших значениях c решения стремятся к своим предельным значениям настолько быстро, что при заданных по умолчанию значениях допустимых ошибок вычислений численная процедура оказывается неспособной выявить детали его поведения при $Z = 250$. Например, при $c = 5$

$$U(250) \approx e^{250\beta} \approx 2.1846 \cdot 10^{-23}.$$

Эта величина слишком мала по сравнению с заданными по умолчанию значениями допустимых ошибок вычислений, однако, если положить `infty=10*c`, мы получим значение

$$U(50) \approx 2.9368 \cdot 10^{-5},$$

которое в большей степени сопоставимо с величиной допустимых ошибок. Если решения дискретной модели становятся численно неразличимыми, матрица соответствующей линейной системы становится вырожденной. При получении приведенного выше сообщения необходимо проверить правильность выбора граничных условий, особенно в случаях наличия в рассматриваемой ЗГУ сингулярности. Проведенный ранее анализ асимптотических граничных условий был обусловлен необходимостью использования аналитического представления решения в тех областях, где трудно выполнить его численную аппроксимацию. В рассматриваемом

[1] Прим. перев. — Перевод на русский сообщения в листинге: «Ошибка при использовании процедуры bvp4c. Невозможно уменьшить шаг сетки — матрица Якоби уравнения коллокации вырождена».

примере мы должны выбрать величину infty настолько большой, что асимптотическое граничное условие адекватно описывает поведение решения, но настолько малой, что выполнение асимптотической аппроксимации остается целесообразным. В контексте результатов, полученных в примере 3.5.4, было бы разумно использовать решение, найденное для одного значения скорости волны, в качестве оценки решения, используемой при получении решения для другого значения скорости волны. Однако, при использовании приведенной ниже программной реализации этого подхода

```
if i == 1
    solinit = bvpinit(linspace(-infty,infty,20),@guess);
else
    solinit = bvpinit(sol,[-infty,infty]);
end
```

можно заметить, что время выполнения вычислений увеличилось до 10.60 секунд. Это не означает, что подход, основанный на использовании продолжения решения по скорости волны, плох; на практике очень часто случается, что единственным «простым» способом вычисления решения на бесконечном интервале является решение этой задачи на интервалах интегрирования, последовательно увеличивающихся на небольшую величину. В рассматриваемой ситуации мы используем оценку решения, имеющую адекватное асимптотическое поведение на одном конце интервала и адекватные качественные характеристики поведения на другом конце интервала. В качестве оценки решения на более длинном интервале используется решение, характеризующееся другим значением волнового числа и заданное на другом интервале, т. е. фактически происходит экстраполяция полученного на предыдущем шаге решения на более длинный интервал. Представленный выше подход с пошаговым увеличением длины интервала интегрирования позволил получить решение задачи, однако, вообще говоря, следовало бы использовать асимптотические выражения для решения, вычисляемые при текущем значении скорости волны, поскольку получаемая при этом аппроксимация решения более точна, чем при использовании подхода с автоматической полиномиальной экстраполяцией численного решения, полученного при другом значении скорости волны.

При вызове численной процедуры интегрирования ЗНУ задаются оценки решения и сетки. При этом важно учитывать, что даже в случае, когда соответствующая процедура для аппроксимации решения позволяет получить это значение для любой точки интервала интегрирования, в численной процедуре используются лишь значения, вычисленные в узлах сетки. Рассмотрим результаты решения задачи Фишера, представленные на рисунке 3.7. На концах интервала интегрирования поведение используемой в программе ch3ex5.m оценки решения полностью соответствует поведению полученного численного решения, но вблизи средней точки этого интервала аппроксимация решения неточна: решение практически сразу становится равным 0 слева от начала координат и равным 1 справа от начала координат! Интересно провести численные эксперименты при различном количестве узлов сетки. Если сетка изначально задается пятью точками, численная процедура оказывается не в состоянии получить решение, поскольку требуемое асимптотическое поведение не выявляется при столь малом количестве узлов сетки. Если сетка

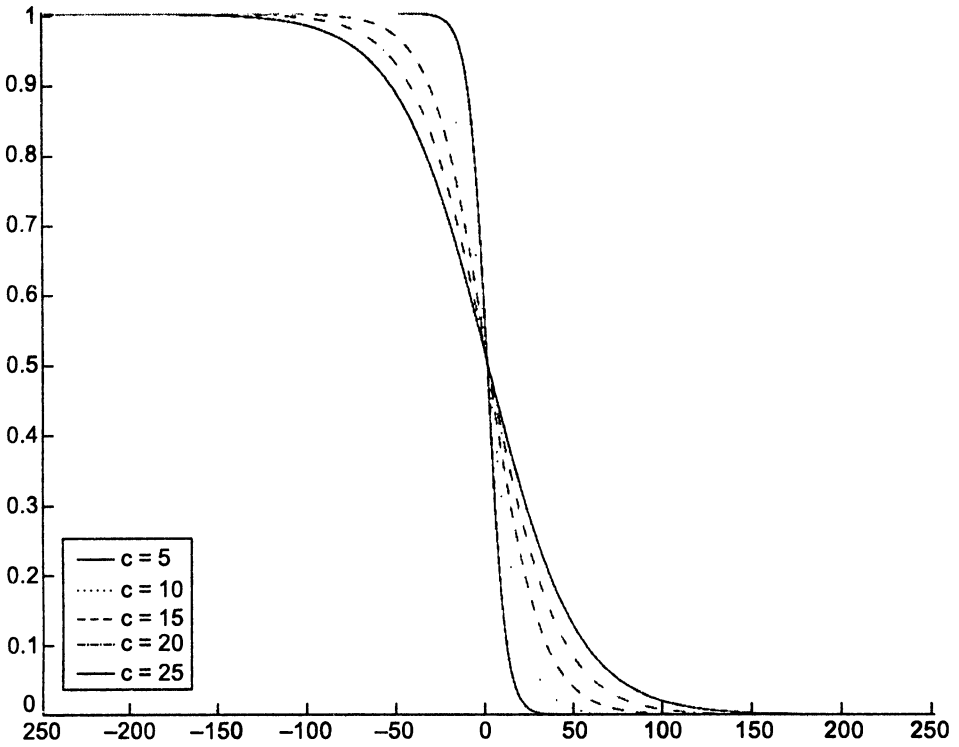

Рис. 3.7. Решения задачи Фишера при различных значениях скорости волны *c*.

определяется 200 равноразнесенными точками, численная процедура позволяет получить решение с требуемыми свойствами, но время вычисления увеличивается с 2.31 секунд до 5.77 секунд. Поскольку вычислительные затраты существенным образом зависят от числа узлов сетки, в численной процедуре `bvp4c` используется лишь столько точек, сколько необходимо. Тем не менее, цель состоит не в минимизации числа используемых точек, а в том, чтобы решить поставленную задачу с заданной точностью. Поэтому алгоритм численной процедуры `bvp4c` разработан таким образом, чтобы исключение неиспользуемых узлов сетки не противоречило общей цели — обеспечению робастности выполняемых вычислений. Если изначально заданные оценки сетки и решения удовлетворительны, ни одна из точек этой сетки не исключается. Вы можете экспериментально определить влияние числа узлов изначально заданной сетки на время выполнения программы. При этом интересно определить число узлов сетки (`length(sol.x)`) для решений, полученных при различных значениях скорости волны *c*. В результате этих экспериментов вы убедитесь в том, что при большинстве значений *c* для вычисления соответствующих решений действительно не требуется сетка с 200 узлами. Другая проблема, связанная с выбором изначальной оценки решения в рассматриваемом примере, заключается в том, что при увеличении числа равноразнесенных узлов изначально заданной сетки увеличивается количество точек этой сетки, расположенных

вблизи начала координат и, следовательно, менее точная аппроксимация решения в этой области может оказывать большее влияние на процесс вычислений. В упражнении 3.26 читателю предлагается использовать асимптотические аппроксимации (по Мюррею), обеспечивающие хорошую оценку решения на всем интервале интегрирования. В упражнениях 3.25, 3.27 и 3.28 читателю предлагается выполнить анализ эффекта от использования векторизации программы и аналитических выражений для вычисления частных производных.

```
function ch3ex5
global c alpha beta infty

options = [];
%options = bvpset(options,'Vectorized','on');
%options = bvpset(options,'FJacobian',@odeJac);
%options = bvpset(options,'BCJacobian',@bcJac);

color = [ 'r', 'b', 'g', 'm', 'k' ];
wave_speed = [5, 10, 15, 20, 25];
hold on
for i = 1:5
    c = wave_speed(i);
    alpha = (-c + sqrt(c^2 + 4))/2;
    beta = (-c + sqrt(c^2 - 4))/2;
    infty = 10*c;
    solinit = bvpinit(linspace(-infty,infty,20),@guess);
    sol = bvp4c(@ode,@bc,solinit,options);
    plot(sol.x,sol.y(1,:),color(i));
    axis([-250 250 0 1]);
    drawnow
end
legend('c = 5', 'c = 10', 'c = 15', 'c = 20', 'c = 25',3);
hold off

%=============================================================
function v = guess(z)
global c alpha beta infty
if z > 0
    v = [exp(beta*z); beta*exp(beta*z)];
else
    v = [(1 - exp(alpha*z)); -alpha*exp(alpha*z)];
end

function dydz = ode(z,y)
global c alpha beta infty
dydz = [ y(2,:); -(c*y(2,:) + y(1,:).*(1 - y(1,:))) ];

function dFdy = odeJac(z,y)
global c alpha beta infty
dFdy = [      0,          1
        (-1 + 2*y(1)), - c ];
```

```
function res = bc(ya,yb)
global c alpha beta infty
res = [ ya(2)/(ya(1) - 1) - alpha
        yb(1)/exp(beta*infty) - 1 ];

function [dBCdya, dBCdyb] = bcJac(ya,yb)
global c alpha beta infty
dBCdya = [ -ya(2)/(ya(1) - 1)^2, 1/(ya(1) - 1)
                  0                      0      ];

dBCdyb = [        0          0
          1/exp(beta*infty), 0 ];
```

ПРИМЕР 3.5.6

В работе [Ascher et al., 1995, пример 1.4] рассмотрена математическая модель процесса впрыска потока жидкости в длинный вертикальный канал. Соответствующая система ОДУ имеет вид

$$\begin{aligned} f''' - R[(f')^2 - ff''] + RA = 0, \\ h'' + Rfh' + 1 = 0, \\ \theta'' + P_e f\theta' = 0, \end{aligned}$$

где R — число Рейнольдса и $P_e = 0.7R$ — число Пекле. Параметр A является неизвестным и рассматриваемая система порядка 7 исследуется при восьми граничных условиях

$$f(0) = f'(0) = 0,\ f(1) = 1,\ f'(1) = 1,$$

$$h(0) = h(1) = 0,\ \theta(0) = 0,\ \theta(1) = 1.$$

Одной из традиционных задач является исследование поведения решения при изменении параметров. При малых отклонениях можно ожидать, что сетка и численное решение (а также неизвестные параметры при их наличии), полученные при одном наборе параметров, являются достаточно хорошими оценками соответствующих величин для другого набора этих параметров. Например, в программе `bvp4c` структура-решение служит в качестве структуры-оценки на следующей итерации. Этот подход представляет собой эффективный способ получения решений ЗГУ при различных наборах параметров и может служить в качестве иллюстрации метода продолжения, часто используемого при решении сложных ЗГУ. Термин «сложная задача» используется здесь в том смысле, что при ее решении трудно предложить изначальную оценку, которая обеспечивала бы сходимость к аппроксимации решения. Рассматриваемая задача является трудной еще и в том смысле, что при большом значении R имеет место пограничный слой, при прохождении через который используется более плотная сетка: устойчивость вычислительного процесса может быть легко обеспечена при значении числа Рейнольдса $R = 100$, но при $R = 10\,000$ возникают проблемы. Метод продолжения позволяет выявить

особенности изменения поведения решения при изменении R и, тем самым, успешно решить ЗГУ при относительно больших значениях R. При решении сложных ЗГУ часто возникает необходимость в большом количестве запусков программы при различных значениях параметров. Более того, запускаемая программа может включать несколько итераций одной и той же процедуры. Поэтому может потребоваться оптимизация алгоритма с целью уменьшения общего времени вычислений. Представленное обсуждение дополняет полученные в примере 3.5.5 результаты, т. к. позволяет выявить отличия, возникающие в ситуациях, когда система содержит неизвестные параметры.

С учетом того, что структура-решение, вычисленная для одного набора параметров, может быть использована в качестве структуры-оценки при вычислении решения для другого набора параметров, рассматриваемая ЗГУ может быть легко решена при нескольких значениях числа Рейнольдса R. При $R = 100$ в качестве оценок компонент решения используются постоянные величины 1, а в качестве оценки неизвестного параметра $A = 1$. Соответствующая структура-оценка определяется с использованием функции `bvpinit`. Решение, полученное для одного значения R, используется при получении решения для другого значения этого параметра. С использованием программы `ch3ex6.m` выводятся графики вычисленных решений (см. рис. 3.8), а также значения A

```
>> ch3ex6
For R = 100, A = 2.76.
For R = 1000, A = 2.55.
For R = 10000, A = 2.49.
```

В представленной версии программы используется векторизация вычислений, а также аналитическое вычисление частных производных.

```
function ch3ex6
global R

color = [ 'k', 'r', 'b' ];
options = bvpset('FJacobian',@Jac,'BCJacobian',@BCJac,...
                 'Vectorized','on');
R = 100;
sol = bvpinit(linspace(0,1,10),ones(7,1),1);
hold on
for i = 1:3
    sol = bvp4c(@ode,@bc,sol,options);
    fprintf('For R = %5i, A = %4.2f.\n',R,sol.parameters);
    plot(sol.x,sol.y(2,:),color(i));
    axis([-0.1 1.1 0 1.7]);
    drawnow
    R = 10*R;
end
legend('R =     100','R =     1000','R =     10000',1);
hold off
```

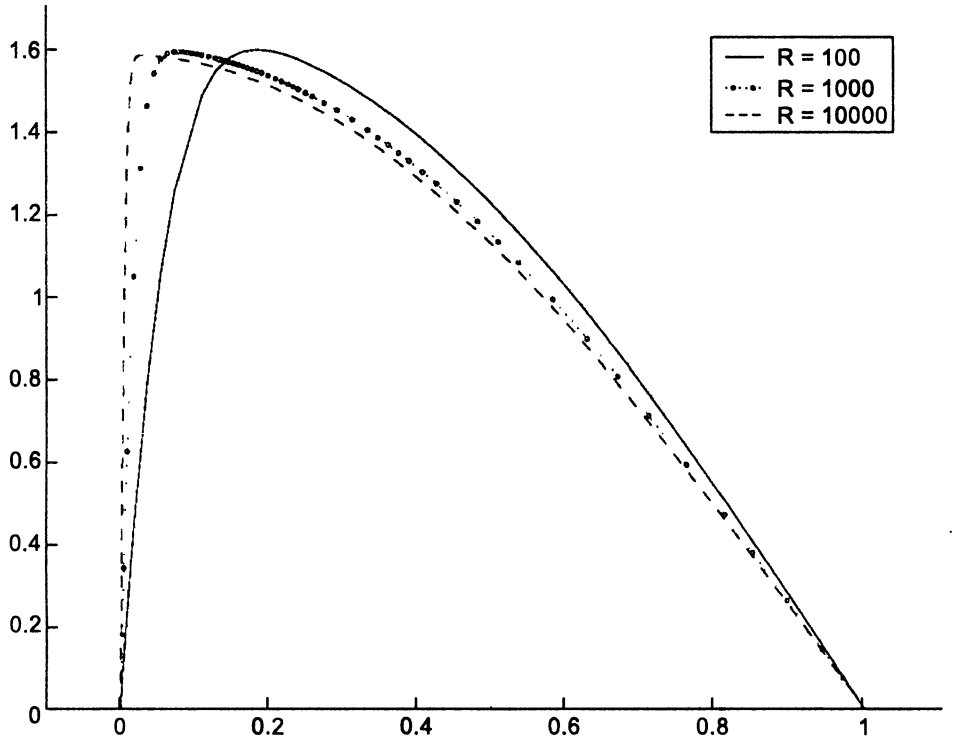

Рис. 3.8. Решение задачи о потоке жидкости, полученное с использованием метода продолжения.

```
%=======================================================================
function dydx = ode(x,y,A);
global R
P = 0.7*R;
dydx = [ y(2,:); y(3,:); R*(y(2,:).^2- y(1,:).*y(3,:)-A);...
         y(5,:); -R*y(1,:).*y(5,:)-1; y(7,:); -P*y(1,:).*y(7,:) ];

function [dFdy,dFdA] = Jac(x,y,A)
global R
dFdy = [     0,        1,        0,      0,    0,      0,       0
             0,        0,        1,      0,    0,      0,       0
          -R*y(3), 2*R*y(2), -R*y(1),  0,    0,      0,       0
             0,        0,        0,      0,    1,      0,       0
          -R*y(5),     0,        0,      0, -R*y(1),  0,       0
             0,        0,        0,      0,    0,      0,       1
        -7/10*R*y(7), 0,        0,      0,    0,      0, -7/10*R*y(1) ];

dFdA = [ 0; 0; -R; 0; 0; 0; 0];

function res = bc(ya,yb,A)
```

```
res = [ ya(1); ya(2); yb(1)-1; yb(2);...
        ya(4); yb(4); ya(6); yb(6)-1 ];

function [dBCdya,dBCdyb,dBCdA] = BCJac(ya,yb,A)
dBCdya = [ 1, 0, 0, 0, 0, 0, 0
           0, 1, 0, 0, 0, 0, 0
           0, 0, 0, 0, 0, 0, 0
           0, 0, 0, 0, 0, 0, 0
           0, 0, 0, 1, 0, 0, 0
           0, 0, 0, 0, 0, 0, 0
           0, 0, 0, 0, 0, 1, 0
           0, 0, 0, 0, 0, 0, 0 ];

dBCdyb = [ 0, 0, 0, 0, 0, 0, 0
           0, 0, 0, 0, 0, 0, 0
           1, 0, 0, 0, 0, 0, 0
           0, 1, 0, 0, 0, 0, 0
           0, 0, 0, 0, 0, 0, 0
           0, 0, 0, 1, 0, 0, 0
           0, 0, 0, 0, 0, 0, 0
           0, 0, 0, 0, 0, 1, 0 ];

dBCdA = zeros(8,1);
```

В отличие от случая, проанализированного в примере 3.5.5, в рассматриваемой здесь задаче имеются неизвестные параметры. Поэтому использование аналитических выражений для частных производных затруднено тем, что необходимо получить частные производные по этим неизвестным параметрам. В программе `ch3ex6.m` идентификаторы для переменных выбраны так, чтобы они указывали на смысл возвращаемого процедурой `Jac` значения. Массив $dFdy$ размерности 7×7 содержит матрицу Якоби для рассматриваемой системы ОДУ. При наличии неизвестных параметров существует второй выходной аргумент процедуры `Jac`, представляющий собой матрицу частных производных системы ОДУ по этим неизвестным параметрам. В рассматриваемом случае система состоит из семи ОДУ и содержит один неизвестный параметр A. Поэтому в качестве массива `dFdA` выступает вектор размерности 7×1. В общем случае, если система состоит из d уравнений и содержит m неизвестных параметров p_j, то матрица частных производных по этим параметрам имеет размерность $d \times m$. На практике эта матрица формируется по столбцам: j-й столбец матрицы представляет собой частную производную вектор-функции f по параметру p_j. В примере 3.5.5 было показано, как функция `jacobian` из пакета символических вычислений MATLAB может быть использована при вычислении матриц частных производных. В рассматриваемом здесь примере для этих целей можно использовать следующую программу

```
syms y y1 y2 y3 y4 y5 y6 y7 R A P F
y = [ y1; y2; y3; y4; y5; y6; y7];
P = 0.7*R;
F = [ y2; y3; R*(y2^2- y1*y3-A); y5;
```

```
      -R*y1*y5-1; y7; -P*y1*y7 ];
dFdy = jacobian(F,y)
dFdA = jacobian(F,A)
```

В результате работы этой программы получается листинг, который после небольшого редактирования может быть использован в качестве основы для написания процедуры `Jac`. Частные производные для граничных условий могут быть получены аналогично с использованием процедуры `BCJac`. Массив `dBCdya` содержит частные производные вектора невязки $g(ya, yb)$ по ya. В рассматриваемом примере мы имеем восемь невязок и вектор ya содержит семь компонент, поэтому массив `dBCdya` имеет размерность 8×7. Массив `dBCdyb` формируется аналогичным образом для вектора yb и имеет ту же размерность. В общем случае, если рассматриваемая система состоит из d уравнений с m параметрами, указанные массивы имеют размерность $(d + m) \times d$. Третьим выходным аргументом процедуры `BCJac` является матрица частных производных невязок по неизвестным параметрам. В рассматриваемом случае имеется только один параметр и поэтому массив `dBCdA` имеет размерность 8×1. В общем случае размерность этого массива $(d + m) \times m$. При решении ЗГУ с использованием аналитических выражений для частных производных пользователь может получить сообщение об ошибке, в котором утверждается, что размерности некоторых массивов, используемых внутри численной процедуры `bvp4c`, несогласованны. На практике это свидетельствует о том, что неправильно определена размерность одной из матриц частных производных. Сравните размерности этих матриц для вашей задачи с размерностями соответствующих матриц, приведенных выше для общего случая. Нередко большинство элементов матриц частных производных равны нулю, и поэтому в программе эти матрицы инициализируются нулями, а ненулевые элементы задаются отдельно. С другой стороны, эти матрицы могут быть сформированы с использованием пакета символьных вычислений, результаты работы которого в несколько отредактированном виде могут составить основу для написания текста соответствующих функций (как это было сделано при написании программы `ch3ex6.m`). На всех этих этапах ручной работы над текстом программы могут быть сделаны ошибки, которые и станут причиной вывода указанного сообщения об ошибке.

Измеряя время выполнения программы аналогично тому, как это делалось в примере 3.5.5, можно установить, что при принятых по умолчанию значениях опций программа `ch3ex6.m` выполняется 13.35 секунд. Вычисление функции в левой части системы ОДУ может быть легко векторизовано: так же как и в примере 3.5.5, это сводится к замене скалярных величин на соответствующие массивы (например, $y(1)$ заменяется на $y(1,:)$) и замене арифметических операций на соответствующие поэлементные операции для массивов (например, `*` и `^` на `.*` и `.^` соответственно). Векторизация ОДУ приводит к уменьшению времени выполнения программы до 8.02 секунд. При использовании аналитических выражений для частных производных, но без векторизации (т. е. при значении опции `Vectorized` равном `off`), рассматриваемая ЗГУ была решена за 6.64 секунд. При векторизации (т. е. при значении опции `Vectorized`, равном `on`) и использовании аналитических выражений для частных производных время выполнения программы сократилось до 4.01 секунд.

Если в качестве значения числа Рейнольдса выбрать $R = 1\,000\,000$, численная процедура bvp4c выдает сообщение[1]

```
Warning: Unable to meet the tolerance without using more
than 142 mesh points.
```

В обычных ситуациях пользователь не должен заботиться о выделении требуемого объема компьютерной памяти, но в численной процедуре установлен предел на максимальное число используемых узлов сетки. Этой величине присваивается значение, равное по умолчанию `floor(1000/d)`, где d — число уравнений в решаемой системе ОДУ. Если численная процедура выдала приведенное выше сообщение, следует увеличить это значение с использованием опции `Nmax`. Указанное сообщение выводится в случаях, когда оценки для сетки и решения недостаточно точны, но увеличение числа точек узлов сетки, предпринятое численной процедурой для обеспечения сходимости вычислительного процесса, не приводит к желаемому результату. Разумеется, вполне возможна ситуация, когда для получения решения с заданной пользователем точностью просто необходимо большее количество узлов сетки. Наконец, возможно, что решения не существует! В рассматриваемом примере решение задачи оказалось возможным лишь при увеличении `Nmax` до 500 и можно сказать, что рассматриваемая задача оказалась достаточно сложной для численной процедуры bvp4c. Действительно, даже при заданных по умолчанию значениях желаемой точности вычислений для надлежащей аппроксимации решения потребовалось 437 узлов сетки. Общее время выполнения программы составило 22.69 секунд с учетом того, что при этом использовались векторизация вычислений и аналитические выражения для вычисления частных производных.

ПРИМЕР 3.5.7

Метод продолжения имеет чрезвычайно важное значение при практическом решении ЗГУ. Ранее мы познакомились с примерами использования метода продолжения по величине длины интервала. Этот подход представляется естественным методом решения задач, определенных на бесконечных интервалах, но может быть применен и при конечных интервалах интегрирования. Мы также рассматривали пример применения метода продолжения по параметрам системы, имеющим физический смысл. При исследовании подобных систем очень часто бывает необходимо найти решения для различных наборов этих параметров, но иногда полезно рассмотреть задачу, которая зависит от единственного изменяющегося параметра и которая может быть легко решена при заданном значении этого параметра. Решение этой задачи может быть выполнено с метода продолжения по параметру. Предположим, что при решении ЗГУ

$$y' = f(x, y),\ 0 = g(y(a), y(b))$$

мы испытываем трудности, обусловленные тем, что мы не можем найти достаточно точные оценки сетки и решения, обеспечивающие сходимость вычислительного процесса. Предположим также, что упрощенная модель ЗГУ

$$y' = F(x, y),\ 0 = G(y(a), y(b))$$

[1] Прим. перев. — Перевод на русский сообщения в листинге: «Предупреждение: невозможно обеспечить требуемую точность вычислений с использованием лишь 142 узлов сетки».

может быть легко решена, возможно, даже аналитически. Часто бывает полезным аппроксимировать исходную задачу соответствующей линейной задачей, поскольку в этом случае численные процедуры типа `bvp4c` наиболее эффективны. Идея заключается в том, чтобы использовать методику продолжения, последовательно решая сначала менее сложную задачу, далее более сложную и, наконец, исходную задачу. Одним из путей реализации этой идеи является введение искусственного параметра μ и решение семейства задач с граничными условиями

$$\begin{aligned} y' &= \mu f(x,y) + (1-\mu)F(x,y), \\ 0 &= \mu g(y(a),y(b)) + (1-\mu)G(y(a),y(b)), \end{aligned}$$

где μ принадлежит промежутку от 0 до 1. Этот подход к решению сложных ЗГУ оказался на практике очень полезным, но, к сожалению, он не всегда приводит к успеху. Очевидно, сформулировать более простую задачу, поведение решений которой было бы адекватно поведению решений исходной задачи, непросто, однако это имеет критическое значение для успешного применения рассматриваемой методики. Более того, метод продолжения в рассматриваемой ситуации может оказаться неработоспособным, поскольку при некотором значении параметра μ соответствующая ЗГУ может просто не иметь решения. В упражнении 3.30 рассматривается именно такой случай. Успех применения этой методики зависит также и от величины изменения значений параметра μ на каждом шаге процесса продолжения.

Методу продолжения посвящена глава 7 в работе [Roberts & Shipman, 1972]. В целях иллюстрации этой методики мы обсудим примеры 1 и 5, рассмотренные в указанной главе. Исследуемая система ОДУ имеет следующий вид

$$\begin{aligned} y_1' &= y_2, \\ y_2' &= y_3, \\ y_3' &= -\left(\frac{3-n}{2}\right) y_1 y_3 - n y_2^2 + 1 - y_4^2 + s y_2, \\ y_4' &= y_5, \\ y_5' &= -\left(\frac{3-n}{2}\right) y_1 y_3 - (n-1) y_2 y_4 + s(y_4 - 1), \end{aligned}$$

где $n = -0.1$ и $s = 0.2$ — параметры. Эта система должна быть решена на промежутке $[0, b]$ с граничными условиями

$$y_1(0) = 0,\ y_2(0) = 0,\ y_4(0) = 0,\ y_2(b) = 0,\ y_4(b) = 1,$$

где $b = 11.3$. Запишем рассматриваемую систему в виде суммы ее линейных членов и нелинейных членов, умноженных на коэффициент $\delta = 1$

$$y' = \begin{pmatrix} y_2 \\ y_3 \\ 1 + s y_2 \\ y_5 \\ s(y_4 - 1) \end{pmatrix} + \delta \begin{pmatrix} 0 \\ 0 \\ -\left(\frac{3-n}{2}\right) y_1 y_3 - n y_2^2 - y_4^2 \\ 0 \\ -\left(\frac{3-n}{2}\right) y_1 y_3 - (n-1) y_2 y_4 \end{pmatrix}.$$

При заданных граничных условиях и при $\delta = 0$ приведенная выше система ОДУ представляет собой линейную аппроксимацию исходной задачи и эта ЗГУ может быть решена при стандартных для подобной ситуации оценках. Далее, решение, полученное при некотором значении δ, используется в качестве оценки при решении ЗГУ с бóльшим значением δ. Эта итеративная процедура должна продолжаться до тех пор, пока не будет достигнуто значение $\delta = 1$, при котором текущая итерация решаемой задачи совпадает с задачей, в решении которой мы были изначально заинтересованы. Программа `ch3ex7.m` представляет собой реализацию этой идеи.

```
function sol = ch3ex7
global delta

sol = bvpinit(linspace(0,11.3,5),ones(5,1));
for delta = [0 0.1 0.5 1]
    sol = bvp4c(@odes,@bcs,sol);
%    plot(sol.x,sol.y)
%    drawnow
%    pause
end
plot(sol.x,sol.y)
axis([0 11.3 -2 1.5])
fprintf('Reference values: y_3(0) = -0.96631, y_5(0) = 0.65291\n')
fprintf('Computed values: y_3(0) = %8.5f, y_5(0) = %8.5f\n',...
         sol.y(3,1),sol.y(5,1))

%================================================
function dydt = odes(t,y)
global delta
n = -0.1;
s = 0.2;
c = -(3- n)/2;
linear = [ y(2); y(3); 1+s*y(2); y(5); s*(y(4) - 1) ];
nonlinear = [ 0; 0; (c*y(1)*y(3) - n*y(2)^2 - y(4)^2); ...
              0; (c*y(1)*y(5) - (n-1)*y(2)*y(4))         ];
dydt = linear + delta*nonlinear;

function res = bcs(ya,yb)
res = [ ya(1); ya(2); ya(4); yb(2); yb(4)-1 ];
```

С использованием этой программы выполняется сравнение вычисленного решения с эталонным решением, представленным в [Roberts & Shipman, 1972]

```
>> sol = ch3ex7;
Reference values: y_3(0) = -0.96631, y_5(0) = 0.65291
Computed values: y_3(0)  = -0.96629, y_5(0) = 0.65293
```

Легко видеть, что численное решение, вычисленное при принятых по умолчанию значениях допустимых ошибок вычисления, достаточно точно совпадает с эталонным решением. Программа позволяет также вывести графики значений всех компонент решения (см. рис. 3.9). Закомментированные строки в тексте программы

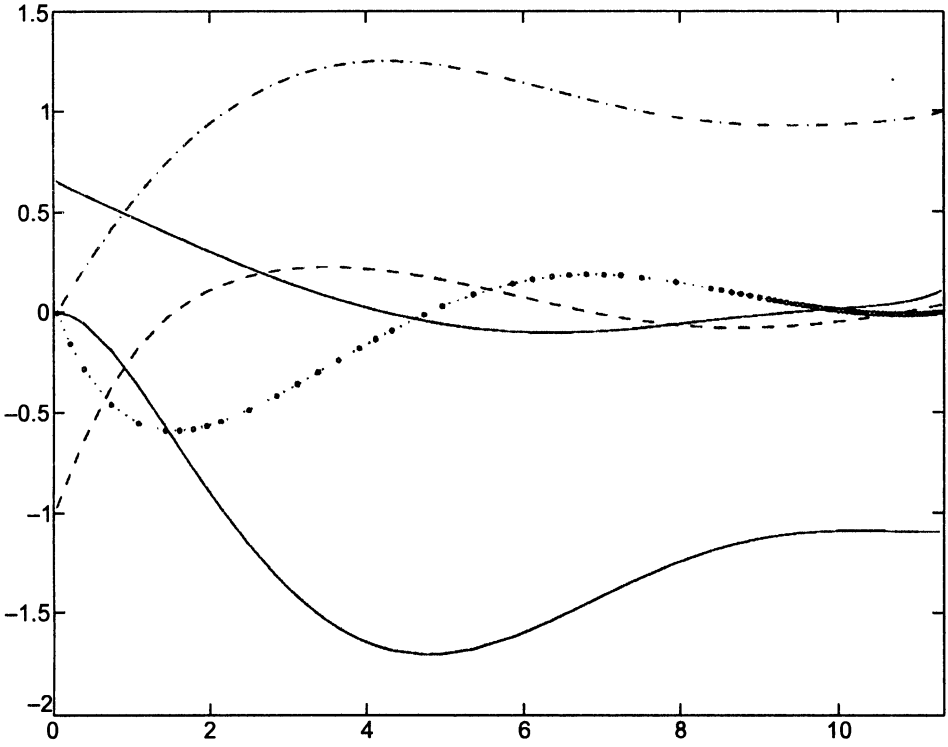

Рис. 3.9. Задача с граничными условиями, решенная методом продолжения по параметру.

`ch3ex7.m` позволяют включить мониторинг изменения решения ЗГУ при последовательном изменении параметра δ. Можно заметить, что линейная аппроксимация рассматриваемой ЗГУ (при $\delta = 0$) не позволяет получить хорошую аппроксимацию решения задачи (при $\delta = 1$), но, тем не менее, является настолько хорошей аппроксимацией, что обеспечивает сходимость вычислительного процесса при относительно большом шаге изменения значения параметра δ в ходе применения методики продолжения.

ПРИМЕР 3.5.8

Некоторые численные процедуры позволяют решать ЗГУ с разделенными граничными условиями, заданными более чем в двух точках, т.е. для многоточечных разделенных граничных условий. Среди них хотелось бы отметить COLNEW [Bader & Ascher, 1987], а также численную процедуру COLSYS [Ascher et al., 1995], снабженную интерфейсом пользователя. Однако большинство численных процедур, включая `bvp4c`, не могут непосредственно использоваться для решения задач с многоточечными разделенными граничными условиями. Пример, рассмотренный в этом параграфе, иллюстрирует метод преобразования подобных ЗГУ к виду, допускающему применение этих стандартных численных процедур. Другие способы достижения этой цели и соответствующие примеры многоточечных ЗГУ можно найти в работах [Ascher & Russell, 1981] и [Ascher et al., 1995, глава 11].

В книге [Lin & Segel, 1988, глава 8] исследовалось движение потока физиологической жидкости. В качестве модели этого процесса использовалась определенная при $0 \leqslant x \leqslant \lambda$ система ОДУ

$$v' = \frac{C-1}{n},$$
$$C' = \frac{vC - \min(x,1)}{\eta},$$

где n и η — известные безразмерные параметры и $\lambda > 1$. Граничные условия имеют вид

$$v(0) = 0,\ C(\lambda) = 1.$$

При исследовании этой задачи наибольший интерес вызывает величина

$$O_s = \frac{1}{v(\lambda)},$$

представляющая собой осмотическую концентрацию раствора. С использованием метода возмущений можно показать, что при малых n справедлива аппроксимация

$$O_s \approx \frac{1}{1-K},$$

где

$$K = \frac{\lambda \sinh(\kappa/\lambda)}{\kappa \cosh(\kappa)}$$

и параметр κ удовлетворяет равенству

$$\eta = \frac{\lambda^2}{n\kappa^2}.$$

Заметим, что член $\min(x,1)$ в уравнении для производной $C'(x)$ не является гладкой функцией в точке $x = 1$. В указанной работе эта ЗГУ рассматривалась как две задачи, определенные соответственно на интервалах $[0,1]$ и $[1,\lambda]$, и связанные друг с другом одним требованием, заключающимся в том, что компоненты решения $v(x)$ и $C(x)$ непрерывны в $x = 1$. В случае, когда ОДУ и, следовательно, его решение не являются гладкими, используемые численные методы характеризуются меньшей устойчивостью. Однако численная процедура `bvp4c` достаточно робастна и с ее помощью эта задача в вышеприведенной формулировке была успешно решена. Для того, чтобы иметь возможность решать подобные задачи в более общей постановке, полезно рассмотреть эту задачу как многоточечную ЗГУ. Действительно, исследуемая задача принадлежит к этому классу, поскольку она включает граничные условия, заданные в трех точках, а не в двух, как в ранее рассмотренных примерах. Переформулировка этой многоточечной задачи в виде двухточечной ЗГУ может быть выполнена следующим образом. Введем две переменные $y_1(x) = v(x)$ и $y_2(x) = C(x)$ и рассмотрим на интервале $0 \leqslant x \leqslant 1$ соответствующую систему дифференциальных уравнений

$$\begin{aligned} \frac{dy_1}{dx} &= \frac{y_2 - 1}{n}, \\ \frac{dy_2}{dx} &= \frac{y_1 y_2 - x}{\eta}. \end{aligned} \tag{3.25}$$

Тогда первое граничное условие принимает вид $y_1(0) = 0$. Далее, введем переменные $y_3(x) = v(x)$ и $y_4(x) = C(x)$ и на интервале $1 \leqslant x \leqslant \lambda$ рассмотрим соответствующие дифференциальные уравнения

$$\frac{dy_3}{dx} = \frac{y_4 - 1}{n},$$
$$\frac{dy_4}{dx} = \frac{y_3 y_4 - 1}{\eta}.$$

Тогда второе граничное условие принимает вид $y_4(\lambda) = 1$. В терминах этих новых четырех переменных условия непрерывности компонент решения v и C могут быть записаны следующим образом: $y_1(1) = y_3(1)$ и $y_2(1) = y_4(1)$. Наиболее трудным этапом решения задачи является одновременное численное решение четырех полученных уравнений. Эта цель может быть достигнута с введением на интервале $1 \leqslant x \leqslant \lambda$ новой независимой переменной

$$\tau = \frac{x - 1}{\lambda - 1}.$$

Подобно независимой переменной x на первом интервале, эта независимая переменная изменяется от 0 до 1 на втором интервале. В терминах этой новой независимой переменной дифференциальные уравнения на втором интервале принимают следующий вид

$$\frac{dy_3}{d\tau} = \frac{(\lambda - 1)(y_4 - 1)}{n},$$
$$\frac{dy_4}{d\tau} = \frac{(\lambda - 1)(y_3 y_4 - 1)}{\eta}. \qquad (3.26)$$

При этом граничное условие $y_4(x = \lambda) = 1$ принимает вид $y_4(\tau = 1) = 1$ и условие непрерывности $y_1(x = 1) = y_3(x = 1)$ переписывается в следующей форме $y_1(x = 1) = y_3(\tau = 0)$. Аналогично, другое условие непрерывности принимает вид $y_2(x = 1) = y_4(\tau = 0)$. Поскольку дифференциальные уравнения для четырех неизвестных связаны между собой граничными условиями, а также учитывая то обстоятельство, что две пары этих уравнений должны быть решены для одной независимой переменной, изменяющейся от 0 до 1, системы ОДУ (3.25) и (3.26) могут быть объединены в одну систему, которую следует решить на интервале $0 \leqslant t \leqslant 1$. В терминах общей независимой переменной t граничные условия имеют вид

$$y_1(0) = 0,\ y_4(1) = 1,\ y_1(1) = y_3(0),\ y_2(1) = y_4(0).$$

То, что граничные условия, обусловленные указанными требованиями непрерывности, являются неразделенными, никак не ограничивает пользователя в случае использования численной процедуры `bvp4c`, однако при использовании других численных процедур, не допускающих решение многоточечных ЗГУ или двухточечных ЗГУ с неразделенными граничными условиями, необходимо предварительно преобразовать исходную задачу так, чтобы разделить граничные условия (см. пример 3.5.3).

Программа `ch3ex8.m` позволяет численно решить рассматриваемую трехточечную ЗГУ при $\kappa = 2,\ 3,\ 4,\ 5$ и $n = 5 \cdot 10^{-2}$, $\lambda = 2$. Решение, полученное при

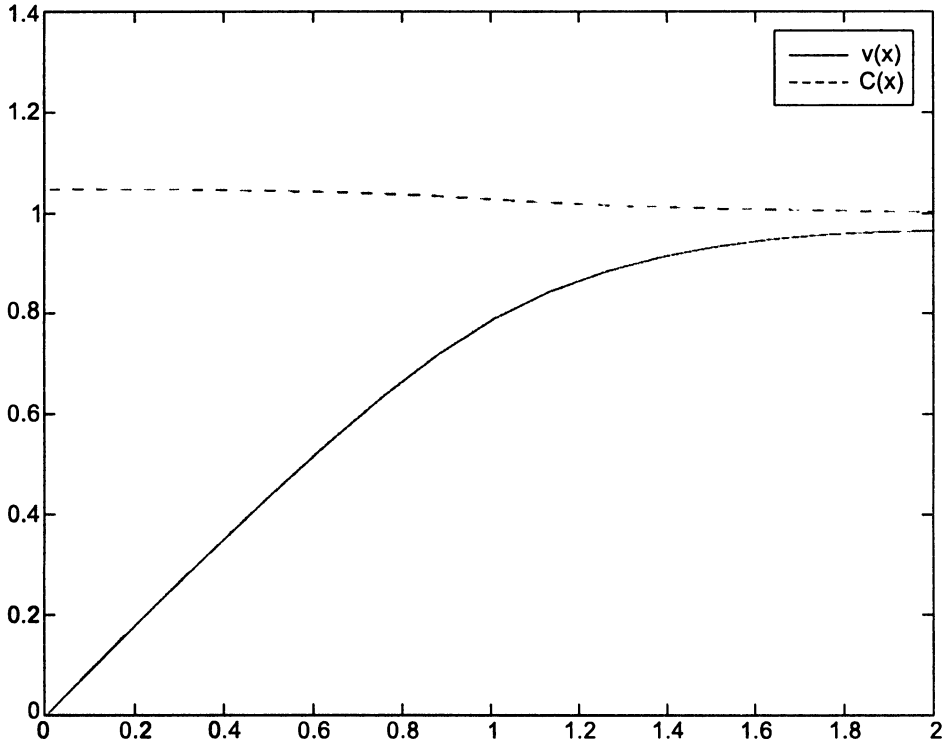

Рис. 3.10. Решение трехточечной ЗГУ.

одном значении κ используется в качестве оценки решения при другом значении этого параметра. Таким образом, этот пример может служить еще одной иллюстрацией использования метода продолжения по физическому параметру системы. Для каждого значения κ вычисленное значение осмолярности O_s сравнивается с приближенным значением, полученным в вышеуказанной работе. Известные параметры передаются в соответствующие функции как дополнительные параметры процедуры `bvp4c`. Поскольку рассматриваемая ЗГУ решается при принятых по умолчанию значениях опций, при вызове `bvp4c` в списке аргументов вместо опций используется `[]`. Для вывода графика решения в исходных переменных необходимо выполнять обратное преобразование переменных на втором интервале и сформировать соответствующее решение на всем интервале $[0, \lambda]$. Программа `ch3ex8.m` выводит значения осмолярности

```
kappa     computed Os     approximate Os
  2         1.462             1.454
  3         1.172             1.164
  4         1.078             1.071
  5         1.039             1.034
```

и графики компонент решения $v(x)$ и $C(x)$ при $\kappa = 5$ (см. рис. 3.10).

```
function ch3ex8
sol = bvpinit(linspace(0,1,5),[1 1 1 1]);
n = 5e-2;
lambda = 2;
fprintf(' kappa computed Os approximate Os\n')
for kappa = 2:5
  eta = lambda^2/(n*kappa^2);
  sol = bvp4c(@odes,@bcs,sol,[],n,lambda,eta);
  K2 = lambda*sinh(kappa/lambda)/(kappa*cosh(kappa));
  approx = 1/(1 - K2);
  computed = 1/sol.y(3,end);
  fprintf(' %2i %10.3f %10.3f\n',kappa,computed,approx);
end
% v and C are computed separately on 0 <= x <= 1 and 1 <= x <= lambda.
% A change of independent variable is used for the second interval.
% First it must be undone to obtain the corresponding mesh in x and
% then a solution assembled for all of 0 <= x <= lambda.
x = [sol.x sol.x*(lambda-1)+1];
y = [sol.y(1:2,:) sol.y(3:4,:)];
plot(x,y(1,:),x,y(2,:))
legend('v(x)','C(x)')

%================================================
function dydx = odes(x,y,n,lambda,eta)
dydx = [ (y(2) - 1)/n
         (y(1)*y(2) - x)/eta
         (lambda - 1)*(y(4) - 1)/n
         (lambda - 1)*(y(3)*y(4) - 1)/eta ];

function res = bcs(ya,yb,n,lambda,eta)
res = [ ya(1); yb(4)-1; yb(1)-ya(3); yb(2)-ya(4)];
```

■ УПРАЖНЕНИЕ 3.15

Покажите, что ЗГУ, определяемая ОДУ

$$y'' + \lambda e^y = 0$$

с граничными условиями $y(0) = 0 = y(1)$, имеет первый интеграл и соответствующий закон сохранения имеет вид

$$(y'(x))^2 + 2\lambda(e^{y(x)} - 1) = (y'(0))^2.$$

Указание: Умножьте ОДУ на $2y'(x)$ и выполните интегрирование. Закон сохранения может быть использован для проверки достоверности вычисленного решения. В ходе обсуждения этого вопроса в параграфе 1.5 было показано, что эти результаты следует интерпретировать осмотрительно. Если численное решение не удовлетворяет закону сохранения, можно с уверенностью утверждать, что оно не является достаточно точным решением рассматриваемой ЗГУ. Однако, даже если численное решение достаточно точно удовлетворяет закону сохранения, нельзя

утверждать, что оно является достаточно точным решением рассматриваемой задачи, т. к. возможна ситуация, когда ошибки вычисления решения коррелируют с законом сохранения таким образом, что последний оказывается выполненным вдоль этого решения. Программа `ch3ex1.m` позволяет вычислить при $\lambda = 1$ приближенные значения решения $y(x)$ и его производной $y'(x)$ в точках x; соответствующие значения записываются в массивы `Sxint(1,:)`, `Sxint(2,:)` и `xint`. С использованием поэлементных векторных операций вычислите и выведите графически значения невязки, которые можно получить путем подстановки численного решения в равенство, определяющее закон сохранения. Удовлетворяет ли вас величина невязки, полученной при интегрировании системы ОДУ со значениями допустимых ошибок вычислений, заданных по умолчанию?

■ УПРАЖНЕНИЕ 3.16

В главе 1 была рассмотрена ЗГУ о движении снаряда. Эта задача имеет два решения, изображенные на рисунке 1.3. Получите этот же график, численно решив систему уравнений, описывающую движение снаряда

$$\begin{aligned} y' &= \operatorname{tg}(\phi), \\ v' &= -\frac{g\sin(\phi) + \nu v^2}{v\cos(\phi)}, \\ \phi' &= -\frac{g}{v^2}. \end{aligned}$$

В качестве значений параметров выберите $g = 0.032$ и $\nu = 0.02$, в качестве интервала интегрирования $[0, 5]$ и в качестве начальной скорости $v(0) = 0.5$. Дополнительные граничные условия определяются требованиями, что начало и конец траектории снаряда расположены на уровне земли, т. е. $y(0) = 0 = y(5)$. В качестве оценок двух решений используйте постоянные значения. Для вывода гладких графиков решений вычислите их значения в точках `linspace(0,5)` при помощи функции `deval`.

■ УПРАЖНЕНИЕ 3.17

В работе [Finlayson, 1972, раздел 5.4] рассмотрена модель трубчатого реактора с осевой дисперсией (axial dispersion). В изотермических условиях при необратимой реакции порядка n эта модель имеет вид следующего ОДУ

$$y'' = P_e(y' + Ry^n),$$

где P_e — осевое число Пекле (axial Peclet number) и R — коэффициент скорости реакции. Граничные условия имеют вид

$$y'(0) = P_e(y(0) - 1), \; y'(1) = 0.$$

В указанной работе с использованием метода ортогональной коллокации было показано, что $y(0) = 0.63678$ и $y(1) = 0.45759$ при $P_e = 1$, $R = 2$ и $n = 2$. Этот результат согласуется с результатами, полученными в других работах с использованием метода конечных разностей. Решите эту задачу, выбрав в качестве оценки сетки десять равноразнесенных на интервале интегрирования точек, а в качестве

оценок $y(x)$ и $y'(x)$ — постоянные значения 0.5 и 0, соответственно. Выведите график численного решения и сравните его значения в точках $x = 0$ и $x = 1$ с приведенным выше результатом, полученным в указанной работе.

■ УПРАЖНЕНИЕ 3.18

В работе [Bailey et al., 1968, пример 5.3] рассматривается длинная, тонкая консольная балка длины L, характеризующаяся жесткостью при изгибе B и подвергающаяся воздействию точечной вертикальной нагрузки P на свободном конце. Смещение балки может быть описано в терминах длины дуги s и угла $\phi(s)$ отклонения балки от горизонтальной оси. Этот угол определяется дифференциальным уравнением

$$\frac{d^2\phi}{ds} + \frac{P}{B}\cos(\phi) = 0$$

с граничными условиями

$$\phi(0) = 0,\ \phi'(L) = 0.$$

В указанной работе решение этой ЗГУ использовалось для исследования некоторой физической величины, но мы предлагаем читателю промоделировать процесс изгиба балки. Для этого к исследуемой системе необходимо добавить уравнения

$$\frac{dx}{ds} = \cos(\phi), \quad \frac{dy}{ds} = -\sin(\phi)$$

с начальными условиями

$$x(0) = 0,\ y(0) = 1.$$

Решите эту ЗГУ при номинальных значениях $L = 10$ и $P/B = 0.001$ и выведите график $(x(s), y(s))$. В качестве оценок решения используйте постоянные значения.

■ УПРАЖНЕНИЕ 3.19

В упражнении 1.9 рассматривался вопрос о представлении в стандартной форме модели гибкой струны, закручивающейся с большой амплитудой [Caughy, 1970]. Здесь мы решим эту задачу численно. В работе [Holmes, 1995, стр. 32–3] было предложено записать решение $\mu(x)$ в виде $\varepsilon y(x)$, после чего с использованием теории возмущений аппроксимировать $y(x)$ при «малых» ε. Функция $y(x)$ удовлетворяет ОДУ

$$y'' + \omega^2\left(\frac{1-\alpha^2}{H}\frac{1}{\sqrt{1+\varepsilon^2 y^2}} + \alpha^2\right)y = 0$$

с граничными условиями

$$y'(0) = 0,\ y'(1) = 0.$$

Физическая константа α удовлетворяет $0 < \alpha < 1$. Поскольку частота вращения ω должна быть определена в ходе решения ЗГУ, возникает другое граничное условие

$$y(0) = 1.$$

Необычным аспектом рассматриваемой задачи является то обстоятельство, что константа H определяется на интервале интегрирования в терминах решения

$$H = \frac{1}{\alpha^2}\left[1 - (1-\alpha^2)\int_0^1 \frac{dx}{\sqrt{1+\varepsilon^2 y^2(x)}}\right].$$

Следуя рекомендациям из упражнения 1.9, сформулируйте эту ЗГУ в стандартной форме. Решите задачу численно при $\alpha = 0.5$ и $\varepsilon = 1$. Аналитически покажите, что при $\varepsilon = 0$ существует решение $y(x) = \cos(\pi x)$ с $\omega = \pi$. Используйте это решение в качестве оценки численного решения при $\varepsilon > 0$. Если бы рассматриваемая задача не была такой простой, для ее решения при $\varepsilon = 1$ потребовалось бы использовать метод продолжения по параметру ε, но вы убедитесь в том, что оценки, полученные как решение задачи при $\varepsilon = 0$, являются достаточно хорошими и обеспечивают устойчивость вычислительного процесса при $\varepsilon = 1$. Выведите график $y(x)$ и значение ω. В работе [Caughy, 1970] аналитически было показано, что $y(0.5) = 0$ и $y(1) = -1$. Определите значения численного решения в этих точках?

■ УПРАЖНЕНИЕ 3.20

В работе [Kubiček, Hlavaček, & Holodnick, 1979] рассматривается модель распределения концентраций и температур в трубчатом реакторе с рециркуляцией, представленная в виде системы ОДУ

$$y' = D_a(1-y)\exp\left(\frac{\gamma\theta}{\gamma+\theta}\right),$$
$$\theta' = BD_a(1-y)\exp\left(\frac{\gamma\theta}{\gamma+\theta}\right) - \beta(\theta-\theta_c)$$

с неразделенными граничными условиями

$$y(0) = (1-\lambda)y(1),\ \theta(0) = (1-\lambda)\theta(1).$$

В указанной работе показано, что при значениях параметров

$$\beta = 0,\ \gamma = 20,\ \lambda = 0.5,\ \theta_c = 1,\ B = 6,\ D_a = 0.05$$

начальные значения удовлетворяют $(y(0), \theta(0)) \approx (0.1, 0.6)$. Подтвердите этот результат и выведите график решения. В указанной работе также показано, что если значение параметра D_a изменить на 0.053, то при тех же значениях остальных параметров существует три решения с начальными значениями $(y(0), \theta(0))$, приблизительно равными $(0.1, 0.7), (0.3, 1.8)$ и $(0.44, 2.6)$. Для получения опыта найдите все эти три решения, используя в качестве оценок компонент решения постоянные значения.

■ УПРАЖНЕНИЕ 3.21

С использованием среды программирования `Mathematica` в работе [Edwards, 1997] исследовалась впервые сформулированная Ньютоном в *Principia Mathematica* задача о выборе «формы тела вращения, обеспечивающей минимальное сопротивление при быстром движении через 'разреженную среду', состоящую из упругих частиц». Эта задача имеет практическое значение, поскольку позволяет выбрать оптимальную форму носового обтекателя самолетов и ракет, движущихся в атмосфере. Соответствующая ЗГУ для получения искомой формы обтекателя может быть сформулирована на основе вариационного исчисления и решена с применением

основанной на методе пристрелки численной процедуры `NDSolve` из пакета `Mathematica`. Соответствующее ОДУ

$$y''(t) = \frac{y'(t)[1 + (y'(t))^2]}{t[3(y'(t))^2 - 1]},$$

рассматривается с граничными условиями

$$y(t_0) = 0,\ y'(t_0) = 1,\ y(1) = 1.$$

В задаче имеется три граничных условия, т. к. $t_0 \in [0, 1]$ выступает в качестве неизвестного параметра. В указанной работе было установлено, что $t_0 \approx 0.3509$. Носовой обтекатель является телом вращения (цилиндром вращения) вокруг вертикальной оси, и при $t_0 > 0$ верхушка обтекателя представляет собой диск радиусом t_0. Кроме того, в работе был определен приведенный коэффициент лобового сопротивления

$$k = t_0^2 + \int_{t_0}^{1} \frac{2\tau}{(y'(\tau))^2 + 1} d\tau,$$

и он оказался равным приблизительно 0.3748. Подтвердите эти два численных результата путем решения задачи с использованием численной процедуры `bvp4c`. Начните исследование с представления рассматриваемой модели в виде системы уравнений первого порядка, введя переменные $y_1(t) = y(t)$ и $y_2(t) = y'(t)$. Далее, так же как в параграфе 1.3, введите для вычисления k новую переменную $y_3(t)$ и добавьте к исследуемой системе соответствующее дифференциальное уравнение, которое будет зависеть только от t и $y_3(t)$. Тогда при $y_3(1) = 1$ после решения ЗГУ вы получите $k = y_3(t_0)$. Далее, аналогично тому, как это было сделано в примере 3.5.3, необходимо сделать замену независимой переменной так, чтобы сформулировать задачу на фиксированном интервале. Пусть x — независимая переменная, изменяющаяся от 0 до 1 на интервале t от t_0 до 1. В терминах этой новой независимой переменной рассматриваемая задача может быть сформулирована на интервале $[0, 1]$, которому принадлежит значение неизвестного параметра t_0. Решите эту ЗГУ с принятыми по умолчанию значениями допустимых ошибок вычислений. В указанной работе было установлено, что $t_0 \approx 0.5$. Отметим, что оценка $y(x) \approx x$ является вполне правдоподобной. После решения ЗГУ выведите график $y(t)$ на интервале $[0, 1]$ — эта кривая представляет собой искомую форму обтекателя. Выведите это решение в виде трехмерной поверхности.

■ УПРАЖНЕНИЕ 3.22

В главе 1 была рассмотрена задача о движении маятника, динамика которого описывается уравнением $\theta'' + \sin(\theta) = 0$ (т. е. уравнением (1.6)) с граничными условиями $\theta(0) = 0$ и $\theta(+\infty) = \pi$. График решения этой задачи изображен на рисунке 1.2 пунктирной линией. Решите эту ЗГУ, заменив граничное условие на бесконечности условием $\theta(T) = \pi$. Для лучшего понимания свойств решений этой задачи решите ее при нескольких значениях T (например, при $T = 5$, 10 и 15). В главе 1 аналитически было показано, что начальный угол наклона $\theta'(0) = 2$. Сравните это значение со значением, полученным при численном решении задачи.

■ УПРАЖНЕНИЕ 3.23

В работе [Cebeci & Keller, 1971] метод пристрелки был использован для получения автомодельного решения задачи Фолкнера–Скана (Falkner–Skan) для вязкого, несжимаемого ламинарного потока, текущего вдоль плоской пластины. Соответствующее ОДУ

$$f''' + ff'' + \beta(1 - (f')^2) = 0$$

решается при граничных условиях

$$f(0) = 0,\ f'(0) = 0,\ f'(+\infty) = 1.$$

Было установлено, что физически приемлемые решения существуют лишь при $-0.19884 \leqslant \beta \leqslant 2$. В указанной работе граничное условие на бесконечности фактически накладывалось в конечной точке. При $\beta > 0$ рассматриваемая ЗГУ может быть непосредственно решена с использованием метода пристрелки, но при некоторых значениях этого параметра приходится использовать метод продолжения. В частности, случай $\beta = 0.5$ оказался наиболее трудным для используемой авторами этой работы численной процедуры. Однако, используя численную процедуру `bvp4c`, вы можете легко решить эту ЗГУ с указанным значением параметра β и граничным условием $f'(6) = 1$. С учетом того, что $f'(x) \approx 1$ на последней части интервала, разумно предположить, что $f(x) \approx x$ и $f''(x) \approx 0$. Покажите с использованием полученного графика решения, что величина $f'(x)$ быстро стремится к 1 при увеличении x. С учетом этого результата можно уверенно утверждать, что в качестве граничного условия на бесконечности можно использовать соответствующее условие, наложенное в конечной точке $x = 6$. Сравните вычисленное вами значение $f''(0)$ со значением 0.92768, полученным в вышеуказанной работе.

■ УПРАЖНЕНИЕ 3.24

Модель вязкого несжимаемого потока за телом полубесконечной длины [Cole, 1968, стр. 159] описывается ЗГУ

$$G' - xF' = 0, \quad G(0) = F(0) = 0,$$
$$F'' + 2(xF - G)F' + 2(1 - F^2) = 0, \quad F(+\infty) = 1.$$

Решите эту задачу, заменив граничное условие на бесконечности на граничное условие $F(X) = 0$, где X — некоторое конечное значение. Для лучшего понимания свойств решения выполните численное интегрирование при нескольких значениях X, например, при $X = 2$, 3 и 4. С физической точки зрения интересна величина трения в пограничном слое, которая кратна $F'(0)$, поэтому выведите вычисленные значения $F'(0)$.

Исследование этой ЗГУ может быть выполнено аналогично тому, как это было сделано в примере 3.8 для другой задачи. Если функция $G(x)$ имеет предел при $x \to \infty$ или, по крайней мере, ограничена, то из граничного условия $F(\infty) = 1$ можно заключить, что $xF - G \sim x$. С учетом этой аппроксимации и, пренебрегая членом $(1 - F^2)$, представляющим собой малую величину при больших x, можно аппроксимировать второе ОДУ следующим уравнением $F'' + 2xF' = 0$. Решая это уравнение, получаем

$$F'(x) \sim \beta e^{-x^2}$$

при постоянном β. Интегрируя и накладывая граничное условие на бесконечности, получаем

$$F(x) \sim 1 - \beta \int^{\infty} e^{-t^2} dt = 1 - \frac{\beta\sqrt{\pi}}{2} \operatorname{erfc}(x).$$

Используя стандартное асимптотическое представление для дополнительной функции ошибок

$$\operatorname{erfc}(x) \sim \frac{e^{-x^2}}{x\sqrt{\pi}},$$

можно показать, что $F(x)$ очень быстро стремится к 1 при увеличении x. Обратившись к рассмотрению первого ОДУ, получаем

$$G'(x) = xF'(x) \sim \frac{\beta}{2}(2xe^{-x^2}).$$

Интегрируя это равенство, имеем

$$G(x) \sim G(\infty) - \frac{\beta}{2} e^{-x^2}.$$

Функция $G(x)$ не только имеет предел при $x \to \infty$, но и стремится к нему очень быстро. Поскольку обе неизвестные функции стремятся при увеличении x к своим пределам очень быстро, мы можем — и в действительности должны — наложить при численном интегрировании граничное условие $F(X) = 1$, которое оказывается выполненным уже при достаточно малом значении X. Эта ЗГУ может быть легко решена численно. Тем не менее, если возникнут какие-либо трудности, вышеприведенный анализ может быть использован для получения более информативных граничных условий. Представленные аналитические аппроксимации точны лишь при больших x, но они могут использоваться при всех x для получения оценки решения, задаваемой при вызове численной процедуры. В этой связи следует отметить, что при определении β и $G(\infty)$ следует наложить граничные условия $G(0) = 0$ и $F(0) = 0$.

■ УПРАЖНЕНИЕ 3.25

Выполните численные эксперименты с использованием программы `ch3ex5.m` в соответствии с указаниями, приведенными в примере 3.5.5. В частности, выясните влияние различных опций на время выполнения этой программы на вашем компьютере. Выполните несколько экспериментов и специально создайте ситуацию, когда численная процедура выдает указанное сообщение об ошибке, чтобы убедиться в том, что численная процедура правильно ее обрабатывает.

■ УПРАЖНЕНИЕ 3.26

В целях аппроксимации решения $U(z)$ задачи Фишера в работе [Murray, 1993] введены новые переменные $\xi = z/c$ и $q(\xi) = U(z)$, в терминах которых ОДУ и граничные условия принимают следующий вид

$$c^{-2}q'' + q' + q(1 - q) = 0,$$
$$q(-\infty) = 1,\ q(\infty) = 0.$$

При «больших» c решение $q(\xi)$ аппроксимируется внешним решением (outer solution) $q_o(\xi)$, т.е. решением уравнения

$$q_o' + q_o(1 - q_o) = 0.$$

В общем случае решение этого уравнения не может удовлетворять обоим граничным условиям, но в рассматриваемой задаче решение имеет вид

$$q_o(\xi) = \frac{1}{1 + e^{\xi}}$$

и действительно удовлетворяет указанным граничным условиям. Возвращаясь к изначальной независимой переменной, переписываем внешнее решение в следующем виде

$$U(z) \approx \frac{1}{1 + e^{z/c}}.$$

В вышеуказанной работе выполнено сравнение этой аппроксимации с численным решением и показано, что она очень точна. Покажите, что вблизи обоих концов аппроксимация имеет адекватные качественные характеристики поведения. В параграфе 3.3.2 было установлено, что (a) при $z \to +\infty$ выполнено $U(z) \sim e^{\beta z}$, где $\beta = (-c + \sqrt{c^2 - 4})/2$ и (b) при $z \to -\infty$ выполнено $U'(z) \sim \alpha(U(z) - 1)$, где $\alpha = (-c + \sqrt{c^2 + 4})/2$. С учетом того, что $q_o \sim e^{-z/c}$ при $z \to +\infty$, покажите, что $\beta \approx -1/c$ при «большом» c. При этом воспользуйтесь аппроксимацией

$$\sqrt{c^2 - 4} = c\sqrt{1 - 4/c^2} \approx c(1 - 2/c^2),$$

полученной с использованием биномиального ряда. Модифицируйте программу `ch3ex5.m` так, чтобы использовать эту аппроксимацию (и ее производную) в качестве оценки. Влияет ли подобная модификация программы на время ее выполнения? Выведите графики аппроксимации и численного решения для некоторого значения c и сделайте заключение о том, насколько хороша эта аппроксимация в качестве оценки решения.

■ УПРАЖНЕНИЕ 3.27

Векторизуйте вычисление ОДУ в программе `ch3ex4.m` и сохраните модифицированный файл под именем `mch3ex4.m`. Оцените влияние векторизации на время выполнения программы, сравнив результат *второго* вызова команды

```
>> tic, mch3ex4, toc
```

при включенной и выключенной векторизацией (т.е. при значениях опции `Vectorized` равных `on` и `off`). Примите во внимание, что при втором и последующих вызовах указанной команды время выполнения имеет сравнимые значения, но может существенно отличаться от времени выполнения при первом вызове этой команды.

■ УПРАЖНЕНИЕ 3.28

Выполните следующие численные эксперименты при решении ЗГУ, рассмотренной в примере 3.5.6. Выясните, как использование векторизации и аналитических

выражений для вычисления частных производных влияет на время выполнения программы. Модифицируйте `ch3ex6.m` для решения рассматриваемой ЗГУ при значении числа Рейнольдса $R = 1\,000\,000$. Что происходит, если задано недостаточно большое число точек сетки. С использованием методики продолжения по параметру R, выполните мониторинг числа узлов сетки `length(sol.x)`, используемых в численной процедуре при численном решении задачи для различных значений R.

■ УПРАЖНЕНИЕ 3.29

В примере 3.5.2 решение соответствующей задачи получено для $\varepsilon = 0.1$, но в работе [Keller, 1992] исследовалось поведение решений при различных значениях этого параметра. В целях увеличения эффективности вычислений в подобной задаче полученное для одного значения ε решение следует использовать в качестве оценки на следующей итерации метода продолжения по параметру ε. Рассматриваемое ОДУ является сингулярным при $\varepsilon \to 0$, т. к. в этом пределе порядок уравнения уменьшается, в результате чего мы получаем алгебраическое уравнение. Решение задач с сингулярными возмущениями, подобных этой, представляет значительные трудности при малых ε, поскольку решение очень быстро изменяется вблизи одного или обоих концов. Это обусловлено тем обстоятельством, что решение соответствующего ОДУ при $\varepsilon = 0$ представляет собой решение ОДУ меньшего порядка, вследствие чего невозможно обеспечить выполнение одновременно всех граничных условий. Рассматриваемая в примере задача необычна тем, что ее решение при $\varepsilon = 0$, имеющее вид

$$y(x) = \sin^2(x),\ \lambda = 1,$$

в действительности удовлетворяет обоим граничным условиям и поэтому пограничные слои отсутствуют. Численная процедура `bvp4c` позволяет легко вычислить решение при малых значениях ε, если в качестве оценки используется это (внешнее) решение. Однако в целях получения практических навыков использования методики продолжения по параметру, вычислите решение задачи при $\varepsilon = 0.01$, выведите его график, а также численно найденное значение неизвестного параметра λ. Для этого модифицируйте программу `ch3ex2.m` так, чтобы последовательно решить рассматриваемую ЗГУ при $\varepsilon = \frac{1}{10}, \frac{1}{20}, \ldots, \frac{1}{100}$. Поскольку ОДУ и граничные условия достаточно просты, нетрудно выписать и использовать аналитические выражения для частных производных. Определите, как это повлияло на время выполнения программы. Команда (в скалярной форме) для вычисления ОДУ в программе `ch3ex2.m` имеет вид

```
dydx = (sin(x)^2 - lambda*sin(x)^4/y)/epsilon;
```

Напомним, что для повышения эффективности вычислений следует векторизовать вышеприведенную команду как по x, так и по y. Эта цель может быть достигнута, например, следующим образом

```
dydx = (sin(x) .^2 - lambda*sin(x) .^4 ./ y(1,:))/epsilon;
```

■ УПРАЖНЕНИЕ 3.30

В работе [Ascher et al., 1995, пример 1.10] рассмотрена модель распространения кори, используемая в качестве иллюстрации методики исследования задач с неразделенными граничными условиями. Периодическое решение системы ОДУ

$$y_1' = \mu - \beta(t) y_1 y_3,$$
$$y_2' = \beta(t) y_1 y_3 - \frac{y_2}{\lambda},$$
$$y_3' = \frac{y_2}{\lambda} - \frac{y_3}{\eta}$$

должно быть найдено на интервале $[0, 1]$. С учетом периодичности решения, оно удовлетворяет равенству

$$y(1) = y(0).$$

Решите эту ЗГУ с использованием численной процедуры `bvp4c`. При использовании численной процедуры, не допускающей решение задачи с неразделенными граничными условиями, выполните разделение этих условий в соответствии с инструкциями, приведенными в вышеуказанной работе. Пусть $c(t)$ — трехмерный вектор. Добавим к системе ОДУ тривиальное уравнение $c' = 0$, решением которого является постоянное решение $c(t) = \text{const}$. Поскольку это решение постоянно, условие периодичности может быть заменено на разделенные граничные условия $y(0) = c(0)$ и $y(1) = c(1)$.

Решите рассматриваемую ЗГУ при $\mu = 0.02$, $\lambda = 0.0279$, $\eta = 0.01$ и $\beta(t) = \beta_0(1 + \cos 2\pi t)$, где $\beta_0 = 1575$. Решение может быть получено с использованием постоянных оценок, но при определении этих констант могут возникнуть сложности. Если вы делаете это упражнение в целях ознакомления с методикой работы с неразделенными граничными условиями, в качестве этих оценок попробуйте использовать вектор `1e-3*ones(3,1)`. Изложенные ниже результаты будут интересны тем читателям, кто желает получить практику в использовании методики продолжения. В примере 3.5.7 система ОДУ аппроксимировалась линейной системой. Однако в рассматриваемом случае возникает проблема, заключающаяся в том, что полученная при этом система ОДУ не имеет периодического решения. В этом можно легко убедиться с использованием линейной аппроксимации первого уравнения $y_1' = \mu$: график решения этого уравнения представляет собой прямую линию с положительным углом наклона. Таким образом, нелинейными членами пренебречь невозможно. С другой стороны, можно ослабить нелинейность так, чтобы полученное уравнение можно было легко решить. Для этого можно рассмотреть β_0 в качестве параметра. При $\beta_0 = 10$ влияние нелинейности невелико и рассматриваемая ЗГУ может быть легко решена с использованием оценки `ones(3,1)`. Используя решение этой ЗГУ в качестве очередной оценки, решите эту же ЗГУ при $\beta_0 = 100$ и выполните следующие итерации метода продолжения по параметру. При $\beta_0 = 10$, 100, 500, 1000 и 1575 мы без труда последовательно вычислили решение. Попытайтесь сделать это при другом выборе последовательности значений параметра β_0. Подобно тому, как это было сделано в примере 3.5.7, выполните мониторинг изменения решения на каждом шаге процедуры продолжения, выводя график при каждом β_0. При выводе графиков компонент решения $y_2(t)$ и $y_3(t)$ полезно масштабировать результат, умножив соответствующие значения на 100.

Глава 4

ДИФФЕРЕНЦИАЛЬНЫЕ УРАВНЕНИЯ С ЗАПАЗДЫВАЮЩИМ АРГУМЕНТОМ

§ 4.1. ВВЕДЕНИЕ

В системе обыкновенных дифференциальных уравнений (ОДУ)

$$y'(t) = f(t, y(t)) \tag{4.1}$$

производная зависит от значения решения в настоящий момент времени t. В системе дифференциальных уравнений с запаздывающим аргументом (ДУЗА) производная зависит также от значений решений в предыдущие моменты времени, т. е. уравнения имеют вид

$$y'(t) = f(t, y(t), y(t-\tau_1), y(t-\tau_2), \ldots, y(t-\tau_k)), \tag{4.2}$$

где запаздывания τ_j — положительные константы, удовлетворяющие неравенствам

$$0 < \tau_1 < \tau_2 < \ldots < \tau_k.$$

Обозначим через τ и $\mathcal{T}$ наименьшее и наибольшее запаздывания. В обзоре [Baker, Paul & Willé, 1995a] приведены многочисленные примеры систем ДУЗА, представляющие собой модели различных систем; в нашей книге мы сконцентрируем наше внимание на биологических моделях. Системы ДУЗА с постоянными запаздываниями образуют обширный и очень важный класс. В работе [Baker, etc., 1995a] приведена всеобъемлющая библиография, касающаяся приложений ДУЗА; в [Baker, Paul & Willé, 1995b] отмечается, что на практике при моделировании систем «в качестве функций запаздывания чаще всего выбираются константы». Это связано, в частности, с тем обстоятельством, что для задач с постоянными запаздываниями можно сравнительно легко разработать программы численного моделирования, которые являются не только более эффективными, но и более робастными по сравнению с соответствующими программами, предназначенными для решения более общих задач. Методы, используемые при численном решении систем ОДУ, в большинстве случаев могут быть обобщены на случай решения систем ДУЗА. В частности, численная процедура MATLAB для решения ДУЗА `dde23` основана на методах, используемых при программной реализации численной процедуры решения ЗНУ `ode23`. Интерфейс пользователя процедуры `dde23` во многом сходен с интерфейсом `ode23`, но имеет некоторые особенности, что обусловлено специфическими отличиями между ДУЗА и ОДУ. Аналогичное сходство имеется между

интерфейсом численной процедуры bvp4c решения ЗГУ и интерфейсом численных процедур, предназначенных для решения ЗНУ.

В параграфе 4.2 рассмотрены принципиальные различия между ДУЗА и ОДУ. В параграфе 4.3 объясняется, как численные методы, разработанные в главе 2 для решения систем ЗНУ, могут быть использованы при решении систем ДУЗА с постоянными запаздываниями. Рассмотренные в параграфе 4.4 примеры иллюстрируют процедуру решения ДУЗА с использованием dde23; кроме того, в этом параграфе более детально обсуждаются отличия между ДУЗА и ОДУ. В заключительном параграфе кратко рассматриваются ДУЗА других типов. При решении всех этих задач возникают определенные трудности, обусловливающие сложность разработки соответствующих численных методов. Тем не менее, программные реализации этих методов (на языке Fortran 77) существуют и ниже они кратко рассматриваются.

§ 4.2. ДИФФЕРЕНЦИАЛЬНЫЕ УРАВНЕНИЯ С ЗАПАЗДЫВАЮЩИМ АРГУМЕНТОМ

Начнем наше изложение с выяснения фундаментальных отличий между задачами с начальными условиями для ДУЗА и ОДУ. Наиболее очевидным отличием является задание начальных данных. Решение системы ОДУ (4.1) определяется его значением в начальной точке $t = a$. С другой стороны, правая часть ДУЗА (4.2) содержит члены вида $y(t - \tau_j)$, которые соответствуют значениям решения, вычисляемым в моменты времени, предшествующие начальной точке. В частности, для вычисления ДУЗА в точке $t = a$ мы должны знать значение $y(a - \mathcal{T})$. Из всего вышеизложенного следует, что в случае решения ДУЗА заданные начальные данные должны включать не только текущее значение решения $y(a)$, но и «историю» изменения этого решения — значения $y(t)$ для всех t из интервала $[a - \mathcal{T}, a]$. Далее численное решение будет обозначаться $S(t)$; при $t \leqslant a$ эта функция используется для определения значений решения в предыстории и называется *функцией истории*.

Поскольку численные методы решения задач с начальными условиями, задаваемыми как для ОДУ так и для ДУЗА, предполагают, что несколько первых производных соответствующих решений непрерывны, нарушения непрерывности производных малого порядка требуют специального анализа. В задачах, определяемых с использованием ОДУ, подобные разрывы имеют место нередко, но в случае решения задач, определяемых с использованием ДУЗА, эта проблема всегда возникает, т. к. первая производная функции истории почти всегда отличается от первой производной решения в начальной точке, т. е. почти всегда выполнено

$$y'(a-) = S'(a-) \neq y'(a+) = f(a, S(a - \tau_1), S(a - \tau_2), \ldots, S(a - \tau_k)).$$

Существуют и другие виды проявления нарушений непрерывности производных малого порядка. Некоторые задачи характеризуются функцией истории, имеющей негладкие производные малого порядка. Например, в упражнении 4.8 мы рассмотрим решение иммунологической модели, рассмотренной в работе [Марчук, 1991]. Одной из компонент функции истории для этой задачи является величина $\max(0, t + 10^{-6})$ для $t \leqslant 0$, т. е. имеет место нарушение непрерывности

первой производной этой компоненты в точке $t = -10^{-6}$. Так же как и в случае решения ОДУ, изменение модели предполагает перезапуск численной процедуры и, следовательно, является нарушением непрерывности первой производной, даже если решение непрерывно в точке смены модели. Это может произойти в заранее известный момент времени, но может определяться и в процессе вычислений посредством аппарата локализации событий. В упражнении 4.7 мы рассмотрим модель распространения инфекции, предложенную Хоппенстедтом и Уолтманом (Hoppensteadt и Waltman). Задача сформулирована на интервале $[0, 10]$. Поскольку для описания различных фаз распространения инфекции используются различные уравнения, можно предполагать, что нарушения непрерывности первой производной решения происходят в заранее известные моменты времени. В примере 4.4.5 рассматриваются дифференциальные уравнения, изменения в которых происходят при возникновении в заранее неизвестные моменты времени некоторого события.

Эффект от нарушения непрерывности может накапливаться в процессе численного нахождения решения и в случае ДУЗА эта проблема имеет более существенное значение, чем в случае решения ОДУ. В этой книге мы не будем выполнять формальный анализ возникающей ситуации, но нетрудно предугадать те процессы, которые возникают в рассматриваемой системе при наличии подобных разрывов. В случае, когда функция f является гладкой, из уравнений (4.2) видно, что гладкость производной y' в текущий момент времени t обусловлена гладкостью решения y в предыдущие моменты времени $t - \tau_j$. Продифференцировав уравнения, можно заключить, что аналогичный вывод можно сделать и в отношении производных старшего порядка. Для лучшего понимания нижеизложенных результатов полезно иметь в виду пример, для которого позднее будет представлено аналитическое решение

$$y'(t) = y(t-1). \tag{4.3}$$

Очевидно, что для этого уравнения справедливо $y^{(k+1)}(t) = y^{(k)}(t-1)$. В общем случае, если в момент времени t^* имеет место нарушение непрерывности (производной) порядка k, т. е. $y^{(k)}$ имеет разрыв в $t = t^*$, то при переходе значения переменной t через $t^* + \tau_j$ возникает нарушение непрерывности функции $y^{(k+1)}$ вследствие наличия в уравнении (4.2) члена $y(t-\tau_j)$. При наличии нескольких запаздываний нарушение непрерывности в момент времени t^* индуцирует нарушение непрерывности и в последующие моменты времени (т. е. распространяется)

$$t^* + \tau_1, t^* + \tau_2, \ldots, t^* + \tau_k,$$

причем каждое из этих нарушений непрерывности также распространяется, т. е. возникает каскад распространения нарушения непрерывности. Если нарушение непрерывности в момент времени t^* имеет порядок k, то нарушение непрерывности в каждый из моментов времени $t^* + \tau_j$ имеет порядок по крайней мере $k+1$, и т. д. Поскольку эффект от наличия запаздывания проявляется при вычислении производной более высокого порядка, в процессе интегрирования решение становится более гладким. Это «сглаживание» имеет очень важное значение при численном интегрировании ДУЗА. Распространение нарушения непрерывности рассматривается более детально в упражнении 4.1. Задачи, в которых возникают подобные нарушения непрерывности, исследованы в упражнениях 4.7, 4.8, 4.10 и 4.14.

Пошаговый метод — это техника решения систем ДУЗА путем их сведения к некоторой последовательности систем ОДУ. Для иллюстрации использования этой методики и для исследования распространения нарушений непрерывности решим уравнение (4.3) с функцией истории $S(t) = 1$ при $t \leqslant 0$. На интервале $0 \leqslant t \leqslant 1$ функция $y(t-1)$ в (4.3) имеет известное значение $S(t-1) = 1$, т. к. $t - 1 \leqslant 0$. На этом интервале рассматриваемое ДУЗА сводится к ОДУ $y'(t) = 1$ с начальным значением $y(0) = S(0) = 1$. Решая эту ЗНУ, получаем $y(t) = t + 1$ при $0 \leqslant t \leqslant 1$. Заметим, что решение ДУЗА демонстрирует типичное нарушение непрерывности первой производной в момент времени $t = 0$, т. к. значение этой производной равно 0 и 1 соответственно слева и справа от начала координат. Поскольку теперь нам известно решение при $t \leqslant 1$, мы можем на интервале $1 \leqslant t \leqslant 2$ свести ДУЗА к ОДУ $y' = (t-1) + 1 = t$ с начальным значением $y(1) = 2$. В результате решения этой новой ЗНУ на указанном интервале получаем решение $y(t) = 0.5t^2 + 1.5$. Заметим, что первая производная непрерывна в $t = 1$, но имеет место нарушение непрерывности второй производной. Нетрудно убедиться в том, что решение рассматриваемого ДУЗА на интервале $[k, k+1]$ имеет вид полинома порядка $k+1$ и что решение характеризуется нарушением непрерывности порядка $k+1$ в момент времени $t = k$. При решении уравнения (4.2) можно использовать полностью аналогичную методику. Если функция истории $S(t)$ определена при $t \leqslant a$, ДУЗА сводится к ОДУ на интервале $[a, a+\tau]$, т. к. при каждом j и при $t - \tau_j \leqslant t - \tau \leqslant a$ величина $y(t-\tau_j)$ определяется известным значением функции истории $S(t-\tau_j)$. Таким образом, мы имеем ЗНУ, определяемую системой ОДУ с начальными данными $y(a) = S(a)$. Решив эту задачу на промежутке $[a, a+\tau]$, можно распространить определение функции $S(t)$ на этот интервал, рассматривая ее как решение этой ЗНУ. Зная решение для $t \leqslant a + \tau$, можно перейти к нахождению решения на интервале $[a+\tau, a+2\tau]$ и т. д. С использованием этой методики решение ДУЗА на всем интервале интегрирования может быть найдено путем решения последовательности ЗНУ, определяемых обыкновенными дифференциальными уравнениями без запаздываний. В этой книге наибольшее внимание будет уделено случаю постоянных запаздываний, но нетрудно понять, что пошаговый метод может быть применен при решении ДУЗА с запаздываниями, зависящими от t и $y(t)$. При этом основным требованием для всех запаздываний является их ограниченность снизу некоторой константой $\tau > 0$.

■ УПРАЖНЕНИЕ 4.1

Следующие упражнения могут использоваться для проверки вашего понимания существа распространения нарушений непрерывности производной, а также пошагового метода решения ДУЗА. При решении этих задач может оказаться полезным пакет компьютерной алгебры системы MATLAB (ядро системы Maple).

- Решите ДУЗА

$$y'(t) = [1 + y(t)]y(t-1)$$

на интервале $1 \leqslant t \leqslant 3$ с функцией истории $y(t) = 1$ при $0 \leqslant t \leqslant 1$. Исследуйте нарушение непрерывности производной в точках $t = 1$ и $t = 2$.

- Решите ДУЗА

$$y'(t) = [1 + y(t)]y(t/2)$$

на интервале $1 \leqslant t \leqslant 4$ с функцией истории $y(t) = 1$ при $\frac{1}{2} \leqslant t \leqslant 1$. Исследуйте нарушение непрерывности производной в точках $t = 1$ и $t = 2$. Определите моменты времени, характеризующиеся тем, что решение непрерывно в этих точках, а производная этого решения терпит разрыв.

- Решите ДУЗА

$$y'(t) = [1 + y^2(t)]y(t-1)$$

на интервале $1 \leqslant t \leqslant 2$ с функцией истории $y(t) = 1$ при $0 \leqslant t \leqslant 1$. Продолжимо ли решение до $t = 2$?

- Добавление небольшого запаздывания в функцию, определяющую правую часть ОДУ, может привести к изменению качественных характеристик его решения. В качестве иллюстрации этого утверждения покажите, что существует конечный момент времени $t^* > 0$, такой что решение ОДУ

$$y'(t) = y^2(t)$$

с начальными данными $y(0) = 1$ не определено при $t \geqslant t^*$. С другой стороны, покажите, что решение ДУЗА

$$y'(t) = y(t)y(t-\tau)$$

с функцией истории $y(t) = 1$ при $t \leqslant 0$ определено при *всех* $t \geqslant 0$ *и при сколь угодно малом запаздывании* $\tau > 0$. Для этого предположите, что $y_k(t)$ является решением ДУЗА на $[k\tau, (k+1)\tau]$. Далее, покажите, что $y_0(t)$ существует и непрерывно при всех $[0, \tau]$. После этого докажите, что если $y_k(t)$ существует и непрерывно при всех $[k\tau, (k+1)\tau]$, то $y_{k+1}(t)$ существует и непрерывно при всех $[(k+1)\tau, (k+2)\tau]$. Таким образом, по индукции решение ДУЗА существует для всех $t \geqslant 0$.

- Все решения ОДУ

$$y'(t) = -y(t)$$

являются убывающими экспонентами. Введение в это уравнение запаздывания может привести к существенному изменению качественных характеристик решения. В частности, докажите, что решения ДУЗА

$$y'(t) = -y\left(t - \frac{\pi}{2}\right)$$

имеют вид $y(t) = A\sin(t) + B\cos(t)$.

§ 4.3. ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ ДУЗА

Системы ДУЗА с постоянными запаздываниями могут быть решены с использованием пошагового метода путем последовательного решения ЗНУ, поставленных для соответствующих систем ОДУ без запаздываний. Этот подход широко распространен вследствие того, что существует множество хорошо известных методов аналитического и численного решения ЗНУ. Как было отмечено выше, в процессе интегрирования решения сглаживаются и поэтому, если минимальная из

имеющихся запаздываний τ мала по сравнению с длиной интервала интегрирования, искомое решение ДУЗА может быть легко найдено путем последовательного решения нескольких сравнительно простых ЗНУ. В подобных условиях явные методы Рунге–Кутта являются удобными и эффективными, вследствие чего большинство численных процедур решения ДУЗА основано именно на этих методах. В частности, численная процедура MATLAB для решения ДУЗА `dde23` основана на BS(2,3)-паре формул, используемой в численной процедуре решения ОДУ `ode23`. Далее мы покажем, как этот подход может быть использован на практике, проиллюстрировав основные идеи на примере реализации численной процедуры `dde23`. Более детальное изложение теоретических и практических аспектов использования `dde23` может быть найдено в работе [Shampine & Thompson, 2001].

Пример, рассмотренный в параграфе 4.2 в качестве иллюстрации пошагового метода, будет использован здесь в целях выявления некоторых проблем, связанных с разработкой методов решения ДУЗА. Напомним, что при нахождении решения уравнения (4.3) на интервале $0 \leqslant t \leqslant 1$ это ДУЗА было сведено к некоторому ОДУ с начальными данными $y(0) = 1$, решение которого характеризовалось тем, что $y(t-1)$ равно заданному значению функции истории $S(t-1)$. Решение этой ЗНУ может быть непосредственно выполнено с использованием явного метода Рунге–Кутта. Однако при переходе к следующему интервалу ситуация усложняется тем, что ОДУ на этом интервале зависит от решения на предыдущем интервале. Заметим, что при использовании классического метода Рунге–Кутта аппроксимация решения может быть вычислена лишь в точках сетки на интервале $[0, 1]$. В первой численной процедуре решения ДУЗА DMRODE [Neves, 1975] для получения аппроксимаций решения в других точках интервала применялась кубическая эрмитова интерполяция. Однако, этот подход не может быть признан удовлетворительным, поскольку размер шага интегрирования, выбираемый с учетом необходимости обеспечения заданной точности интегрирования, может оказаться недостаточно малым для точной интерполяции. В подобных ситуациях естественным решением проблемы является использование непрерывного обобщения. Для метода Рунге–Кутта, основанного на BS(2,3)-паре формул и используемого в численной процедуре `ode23`, с применением эрмитовой интерполяции может быть разработано непрерывное обобщение.

При достижении в процессе вычислений текущего момента времени t мы должны иметь возможность вычислить аппроксимацию решения $S(t)$ на основании информации, накопленной на предыдущих шагах интегрирования, выполненных за последний отрезок времени длиной $t-\mathcal{T}$. Это означает, что необходимо предусмотреть возможность сохранения информации, необходимой для вычисления кусочно-полиномиальной функции $S(t)$. Поскольку непрерывное обобщение для BS(2,3)-пары эквивалентно кубической эрмитовой интерполяции между узлами сетки, достаточно хранить значения точек сетки, а также значение аппроксимации решения и его производной, вычисленные в каждой из этих точек сетки. Численная процедура `dde23` возвращает структуру-решение, для которой мы выбрали идентификатор `sol`. Тогда узлы сетки, значения решения и его производной в узлах сетки могут быть получены как поля структуры-решения `sol.x`, `sol.y` и `sol.yp` соответственно. Подобная форма представления выходных переменных является опциональной для численной процедуры `ode23`, но единственно возможной для `dde23`.

Так же как и в случае численных процедур решения ЗНУ, непрерывное обобщение может быть вычислено функцией `deval` с использованием информации, содержащейся в структуре-решении. Часто бывает полезным вычислить решение в любой точке интервала интегрирования, но в отличие от случая применения численных процедур решения стандартных ЗНУ, в рассматриваемой ситуации наличие этой возможности является необходимым условием для решения задачи.

Представление решения в форме структуры упрощает интерфейс с пользователем. Далее мы увидим, что при решении ДУЗА наряду с данными, необходимыми для выполнения интерполяции, необходимо иметь и другую информацию, содержание которой зависит от специфики рассматриваемой задачи. Хранение этой информации в виде полей структуры-решения удобно для пользователя и не загромождает текст программы операторами, имеющими лишь техническое значение. Ранние версии численных процедур решения ДУЗА были написаны на языке Fortran, в реализации которого отсутствует возможность динамического выделения компьютерной памяти. Это существенно усложняло интерфейс с пользователем и нередко приводило к тому, что выделенного объема оказывалось недостаточно для полного решения задачи. Динамическое выделение компьютерной памяти, предусмотренное в MATLAB при использовании структур, позволило упростить и усовершенствовать интерфейс пользователя численной процедуры `dde23`.

Рассмотренное ранее ДУЗА (4.3) сводится к легко интегрируемому ОДУ и поэтому в целях повышения эффективности вычислений применяемая численная процедура использует максимальный шаг интегрирования. Действительно, формулы Рунге–Кутта являются точными на первом интервале, но интервал интегрирования ограничен сверху точкой $t = 1$, поскольку в этой точке решение не является гладким. Если не принимать во внимание факт нарушения непрерывности, можно использовать формулы Рунге–Кутта меньшего порядка. В этом случае полученное численное решение будет не столь точным, как нам бы того хотелось, но самым большим недостатком подобного пути решения задачи является то, что в этом случае стандартный критерий оценки ошибки вычислений становится неприменимым. Это обстоятельство обусловлено тем, что эта ошибка оценивается путем сравнения результатов вычисления двух формул и в случае, когда функция f в правой части дифференциального уравнения не является достаточно гладкой, порядок каждой из этих формул не соответствует стандартной ситуации. Эта проблема может быть решена путем подбора шага интегрирования, выполненного таким образом, чтобы все точки, в которых решение $y(t)$ имеет нарушение непрерывности производных малого порядка, являлись бы узлами сетки. Тогда ни одна из функций $y(t)$, $y(t-\tau_1), \ldots, y(t-\tau_k)$ не будет иметь нарушения непрерывности производных малого порядка на протяжении отдельного шага от точки t_n до t_n+h. Поскольку численное решение $y(t)$ определяется в узлах сетки, с которыми совпадают точки нарушения непрерывности, аналогичное утверждение можно сделать и в отношении самого решения $y(t)$: не существует точки ξ, принадлежащей $(t_n, t_n + h)$, в которой некоторая функция $y(\xi - \tau_j)$ является разрывной, поскольку наличие разрыва функции $y(t)$ в точке $\xi - \tau_j$ привело бы к возникновению нарушения непрерывности в точке ξ и в этом случае нам бы пришлось уменьшить шаг интегрирования h так, чтобы эта точка совпала с одним из новых узлов сетки. Формулы Рунге–Кутта являются одношаговыми и поэтому, если мы будем следовать представленной методике, эти

формулы будут применяться в отношении ОДУ с функцией в правой части, являющейся гладкой на интервале шага интегрирования, и, следовательно, порядок этих формул будет равен стандартным значениям, соответствующим ситуации, когда правая часть ОДУ является достаточно гладкой функцией.

Как отмечалось ранее, указанные нарушения непрерывности представляют собой существенную трудность при решении ДУЗА, поскольку ситуации, когда это происходит в начальной точке, нередки, и, кроме того, это нарушение непрерывности распространяется на весь интервал интегрирования. С другой стороны, поскольку на каждом шаге процесса распространения порядок нарушений непрерывности возрастает, их анализ следует выполнять лишь с точки зрения их влияния на возможность применения в соответствующих ситуациях выбранного численного метода. Перед тем, как начать интегрирование, численная процедура `dde23` определяет точки нарушений непрерывности, порядок которых достаточно мал для того, чтобы повлиять на выполнение вычислений. При этом предполагается, что в начальной точке имеет место нарушение непрерывности первой производной. В некоторых задачах заранее известны дополнительные точки, в которых имеют место нарушения непрерывности. Эти точки передаются в численную процедуру `dde23` с использованием опции `Jumps`. Значения опций могут быть заданы с использованием дополнительной функции `ddeset`, аналогично тому, как задаются опции численных процедур решения ЗНУ с использованием функции `odeset`. Например, три точки нарушения непрерывности в модели Хоппенстедта–Уолтмана из упражнения 4.7 могут быть заданы следующим образом

```
c = 1/sqrt(2);
options = ddeset('Jumps',[(1-c), 1, (2-c)]);
```

Нарушения непрерывности в предыстории задаются аналогично тому, как это делается в отношении разрывов в точках, принадлежащих остальной части интервала интегрирования (см. упражнение 4.8, в котором исследуется иммунологическая модель). В упражнении 4.10 читателю предлагается попрактиковаться в использовании опции `Jumps`. Иногда начальное значение $y(a)$ отличается от соответствующего значения функции истории $S(a)$. В подобной ситуации значение $y(a)$ передается в процедуру как значение опции `InitialY`. Ситуация, когда в начальной точке имеет место нарушение непрерывности самого решения, обрабатывается аналогично стандартной ситуации, но с учетом того, что его порядок увеличился на единицу. Опция `InitialY` используется при решении ДУЗА в упражнении 4.6.

При распространении каждого нарушения непрерывности в точке ξ его порядок увеличивается на единицу в каждой из точек

$$\xi + \tau_1, \xi + \tau_2, \ldots, \xi + \tau_k.$$

При этом каждое из полученных таким образом нарушений непрерывности в свою очередь индуцирует нарушение непрерывности и распространяется аналогичным образом. Точки нарушений непрерывности образуют древовидную структуру, которая может быть усечена, если порядок нарушения непрерывности настолько большой, что это нарушение непрерывности не оказывает воздействия на точность вычислений с использованием выбранного численного метода. Тем не менее,

в связи с распространением нарушений непрерывности могут возникнуть определенные трудности, существо которых может быть продемонстрировано на следующем примере. Предположим, что ДУЗА имеет два запаздывания $\frac{1}{3}$ и 1 и интегрирование начинается при $t = 0$. Первое запаздывание обусловливает возникновение нарушений непрерывности в точках 0, $\frac{1}{3}$, $2 \times \frac{1}{3}$, $3 \times \frac{1}{3}$, ... а второе — в точках 0, 1, 2, 3, ... Проблема заключается в том, что представление величины $3 \times \frac{1}{3}$ с конечной точностью не равно в точности единице. Таким образом, численная процедура имеет две численно неразличимых точки нарушения непрерывности. При непосредственном выполнении вычислений это обстоятельство должно привести к катастрофическим последствиям, т. к. шаг интегрирования ограничен абсолютной величиной разности между точками нарушения непрерывности. В численной процедуре dde23 подобная ситуация обрабатывается следующим образом: если два значения, задающих точки нарушения непрерывности, отличаются друг от друга менее чем на десятикратную величину ошибки округления, одно из значений игнорируется. Для того, чтобы как можно раньше удалить одну из численно неразличимых точек, выявление подобной ситуации производится на каждом шаге распространения.

Решение системы ДУЗА (4.2) сглаживается в процессе интегрирования, вследствие чего мы можем ожидать увеличение шага интегрирования. Разумеется, мы должны ограничить рост шага интегрирования во избежание пропуска момента нарушения непрерывности производных малого порядка. Но что делать в ситуациях, когда после некоторого момента времени подобные точки отсутствуют? В этом случае шаг интегрирования остается ограниченным величиной минимального запаздывания и поэтому при интегрировании уравнения на интервале от t_n до $t_n + h$ с шагом $h > \tau$, по крайней мере один из аргументов $t - \tau_j$ окажется в интервале $(t_n, t_n + h]$. Таким образом оказывается, что нам необходимо иметь значения решения в точках, расположенных в интервале шага интегрирования. Однако мы еще только пытаемся вычислить решение на этом интервале и поэтому не знаем этих значений! В некоторых численных процедурах эта проблема решается тем, что шаг интегрирования остается ограниченным. В других численных процедурах, включая dde23, выбирается какой-либо шаг интегрирования, обеспечивающий гладкость решения, и выполняется итеративное вычисление возникающих при этом неявных формул. При достижении t_n мы имеем кубическую кусочно-полиномиальную аппроксимацию $S(t)$ решения при $t \leqslant t_n$. Значения $y(t - \tau_j)$ при $t - \tau_j > t_n$, необходимые при вычислении BS(2,3)-формул, определяются с использованием экстраполяции полиномиальной аппроксимации решения, полученной на предыдущем интервале. После вычисления формул мы получаем новую кубическую полиномиальную аппроксимацию решения на интервале $(t_n, t_n + h]$, которая используется для коррекции решения путем перевычисления формул. Вычисление BS(2,3)-формул при шаге интегрирования большем, чем минимальная из величин запаздывания, выполняется аналогично вычислению формул неявного многошагового метода.

Ранее мы упоминали об использовании аппарата локализации событий при изменении исследуемой модели. Эта возможность может быть использована в численной процедуре dde23 аналогично тому, как это делается в численной процедуре ode23 с учетом лишь одного дополнительного обстоятельства — информация

о событиях всегда возвращается в виде полей структуры-решения. При возникновении терминального события интегрирование обычно продолжается, но с модифицированными уравнениями и новым начальным значением, в качестве которого используется (возможно модифицированное) конечное значение решения, полученное на предыдущем этапе интегрирования. Описанная процедура проста в случае численного решения ОДУ с использованием процедур решения ЗНУ системы MATLAB, т. к. решения этой последовательности задач могут быть легко агрегированы в единое решение на всем интервале интегрирования. Однако при решении ДУЗА возникает совершенно другая ситуация. Основная трудность заключается в том, что для интегрирования на последующем этапе необходимо задать историю. В большинстве случаев эта история может быть задана с использованием решения, полученного на предыдущем этапе интегрирования до момента возникновения события, но она может также включать историю, используемую на текущем этапе интегрирования. При использовании `dde23` аргумент истории может задаваться тремя различными способами (более детально это будет рассмотрено ниже на конкретном примере). Численная процедура `dde23` использует информацию, содержащуюся в структуре-решении, для определения истории и вычисления зависящих от запаздываний членов в ДУЗА. Другая потенциальная проблема связана с распространением нарушения непрерывности. Возможна ситуация, когда событие возникает в момент, когда некоторые распространяющиеся нарушения непрерывности находятся в процессе обработки в численной процедуре. На текущем этапе интегрирования мы должны перестроить дерево нарушений непрерывности и распространить его на текущий интервал интегрирования. Для этого необходимо располагать некоторой информацией, полученной и сохраненной на предыдущем этапе интегрирования. Разумеется, всегда возникает новое нарушение непрерывности, обусловленное заданием новой начальной точки. Нередко случается, что начальное значение решения должно отличаться от финального значения решения на предыдущем этапе интегрирования. Эта проблема может быть решена с использованием опции `InitialY`. В случае, когда для задания истории используется структура-решение, численная процедура `dde23` агрегирует эту структуру с новой структурой-решением, получаемой на текущем этапе интегрирования. Таким образом, структура-решение, возвращаемая численной процедурой на заключительном этапе, всегда содержит информацию о решении на всем интервале интегрирования.

§ 4.4. РЕШЕНИЕ ДУЗА В СИСТЕМЕ MATLAB

При использовании численных процедур решения ДУЗА пользователь должен задать больше информации, чем в случае решения ОДУ, т. к. ДУЗА являются уравнениями более общего вида. Несмотря на то, что детали этой информации могут отличаться для различных численных процедур, существует некоторый базисный набор данных, который необходим при использовании всех этих процедур. В этом параграфе мы покажем, как численная процедура `dde23` может быть использована при решении ДУЗА. Область применения этой процедуры ограничена случаем, когда запаздывания рассматриваемого ДУЗА постоянны. Приведенные ниже примеры и упражнения показывают, что для указанного класса задач `dde23` является

очень эффективной. Решение ДУЗА с использованием `dde23` осуществляется во многом аналогично решению ОДУ с использованием `ode23`, но имеются некоторые существенные отличия. Некоторые их рассмотренных ниже примеров могут рассматриваться как тесты работоспособности численных процедур решения ДУЗА.

ПРИМЕР 4.4.1

В работе [Genik & van den Driessche, 1999] рассматривается эпидемическая модель с дискретными запаздываниями, в которой вся популяция с общей численностью $N(t)$ разделена на четыре группы: $S(t)$ — восприимчивые к болезни, $E(t)$ — имеющие контакт с болезнетворными микробами, но неинфицированные, $I(t)$ — инфицированные (т. е. фактически заболевшие), и $R(t)$ — выздоровевшие после болезни. Таким образом, общая численность популяции определяется равенством

$$N(t) = S(t) + E(t) + I(t) + R(t).$$

Пять параметров модели, построенной для случая вируса пастереллеза *Pasteurella muris*, введенного в организм лабораторных мышей, следующие: рождаемость $A = 0.330$, естественная смертность $d = 0.006$, коэффициент контакта инфицированных особей $\lambda = 0.308$, коэффициент выздоровления $\gamma = 0.040$, коэффициент сверхнормативной смертности заболевших особей $\varepsilon = 0.060$. В модели имеются два запаздывания: временно́е запаздывание невосприимчивости к болезни $\tau = 42.0$ и инкубационный период (период времени между моментом заражения и моментом начала заболевания), $\omega = 0.15$. Модель описывается следующей системой ДУЗА:

$$\begin{aligned}
\frac{dS(t)}{dt} &= A - dS(t) - \lambda\frac{S(t)I(t)}{N(t)} + \gamma I(t-\tau)e^{-d\tau},\\
\frac{dE(t)}{dt} &= \lambda\frac{S(t)I(t)}{N(t)} - \lambda\frac{S(t-\omega)I(t-\omega)}{N(t-\omega)}e^{-d\omega} - dE(t),\\
\frac{dI(t)}{dt} &= \lambda\frac{S(t-\omega)I(t-\omega)}{N(t-\omega)}e^{-d\omega} - (\gamma+\varepsilon+d)I(t),\\
\frac{dR(t)}{dt} &= \gamma I(t) - \gamma I(t-\tau)e^{-d\tau} - dR(t).
\end{aligned}$$

Эти уравнения должны быть решены на интервале $[0, 350]$ с учетом истории, определенной следующими равенствами: $S(t) = 15$, $E(t) = 0$, $I(t) = 2$ и $R(t) = 3$ для всех моментов времени $t \leqslant 0$. Аналитическое исследование показывает, что существует граница области устойчивости $R_0 = \lambda e^{-d\omega}/(\gamma+\varepsilon+d)$. Эпидемия прекратится, если $R_0 < 1$, и будет развиваться, если $R_0 > 1$. При вышеприведенных значениях параметров $R_0 \approx 2.9$.

Процедура численного решения рассматриваемой системы ДУЗА с использованием `dde23` полностью аналогична процедуре решения ЗНУ с использованием `ode23`. Поэтому далее мы лишь представим листинг программы и обсудим различия. Результаты вычислений представлены в графическом виде на рисунке 4.1.

```
function sol = ch4ex1
global tau omega
tau = 42.0; omega = 0.15;
```

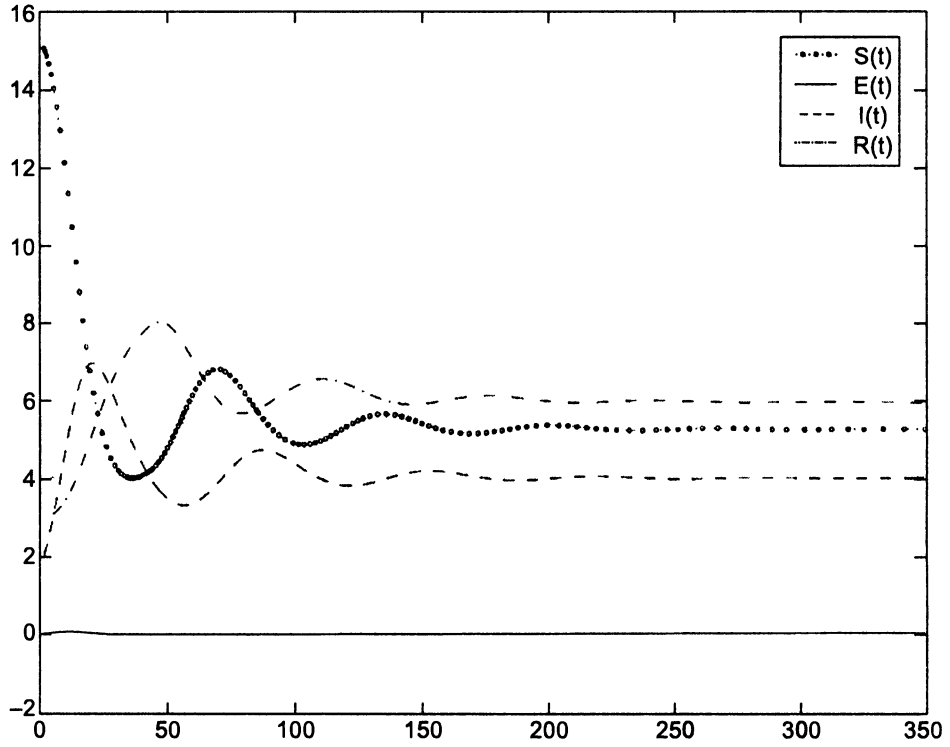

Рис. 4.1. Модель эпидемии SEIRS [Genik & van den Driessche, 1999].

```
sol = dde23(@ddes, [tau, omega], [15; 0; 2; 3], [0, 350]);
plot(sol.x,sol.y)
legend('S(t)', 'E(t)', 'I(t)', 'R(t)')

%=========================================
function dydt = ddes(t,y,Z)
global tau omega

% Parameters:
A = 0.330; d = 0.006; lambda = 0.308;
gamma = 0.040; epsilon = 0.060;

% Variable names used in stating the DDEs:
S = y(1); E = y(2); I = y(3); R = y(4);
% Z(:,1) corresponds to the lag tau.
Itau = Z(3,1);
% Z(:,2) corresponds to the lag omega.
Somega = Z(1,2); Eomega = Z(2,2);
Iomega = Z(3,2); Romega = Z(4,2);
```

```
Noft = S + E + I + R;
Nomega = Somega + Eomega + Iomega + Romega;
dSdt = A - d*S - lambda*((S*I)/Noft) + gamma*Itau*exp(-d*tau);
dEdt = lambda*((S*I)/Noft) - ...
       lambda*((Somega*Iomega)/Nomega)* exp(-d*omega) - d*E;
dIdt = lambda*((Somega*Iomega)/Nomega)*exp(-d*omega) ...
       - (gamma+epsilon+d)*I;
dRdt = gamma*I - gamma*Itau*exp(-d*tau) - d*R;

dydt = [ dSdt; dEdt; dIdt; dRdt];
```

Вызов процедуры dde23 имеет вид `sol = dde23(@ddes,lags,history,tspan)`. Так же как и в случае использования численных процедур решения ЗНУ, аргумент tspan представляет собой интервал интегрирования, но используется этот массив несколько по-другому. Если tspan содержит более двух элементов, численная процедура решения ЗНУ возвращает значения решения, вычисленные в этих точках; в численной процедуре dde23 используемыми элементами этого массива являются лишь первый и последний, остальные игнорируются. Кроме того, если tspan равен $[t_0, t_f]$, для успешного вызова dde23 необходимо $t_0 < t_f$. Аргумент history может задаваться в трех формах. В качестве одного из возможных вариантов может выступать указатель на функцию, в которой вычисляется значение решения в заданной точке $t \leqslant t_0$ с возвращением полученного результата в виде вектора-столбца. В рассматриваемом примере эта функция может иметь следующий вид

```
function v = history(t)
v = [15; 0; 2; 3];
```

Функция истории часто является постоянной, и поэтому соответствующий аргумент численной процедуры можно задавать в виде постоянного вектора, что и было сделано в программе ch4ex1.m. (Третьей формой представления аргумента истории является структура-решение. Эта форма используется лишь в тех случаях, когда необходимо выполнить дополнительный этап интегрирования. В рассматриваемом примере это не делается.) Запаздывания передаются в процедуру в виде вектора lags (в рассматриваемом примере в качестве этого вектора выступает [tau,omega]). Аналогично тому, как функция odes используется для вычисления ОДУ, функция ddes используется для вычисления ДУЗА. Кроме того, так же как и в случае решения ОДУ, выходной аргумент t содержит текущее значение t, а входной аргумент y — аппроксимацию решения $y(t)$. Единственным отличием случая решения ДУЗА является наличие дополнительного входного массива Z. Этот массив содержит значения аппроксимаций решения при всех запаздываниях. В частности, Z(:,j) содержит аппроксимации функции $y(t - \tau_j)$ при величине запаздывания τ_j, значение которой записано в элементе массива lags(j). Аналогично случаю решения ОДУ, функция ddes возвращает вектор-столбец.

Выходным аргументом численной процедуры dde23 является структура-решение, для которой в рассматриваемом примере выбран идентификатор sol. Поля этой структуры sol.x, sol.y и sol.yp содержат, соответственно, сетку узлов, значения решения и его производных в этих узлах. Как было отмечено ранее, эта

форма вывода является опциональной для численной процедуры ode23, но единственной формой вывода для dde23. Вы можете вызвать программу ch4ex1.m без указания выходных аргументов, но если вы желаете выполнить дополнительное исследование полученного решения, этот вызов следует выполнить следующим образом

```
>> sol = ch4ex1;
```

После этого вы можете, например, вычислить и вывести график изменения общей численности популяции $N(t)$, используя сохраненную в структуре sol информацию. Это может быть сделано следующим образом

```
>> Noft = sum(sol.y);
>> plot(sol.x,Noft)
```

ПРИМЕР 4.4.2

В этом примере мы покажем, как можно вывести результаты вычислений для точек, заданных специфическим образом. Рассмотрим следующее скалярное уравнение, демонстрирующее хаотическое поведение [Willé & Baker, 1992, пример 5]

$$y'(t) = \frac{2y(t-2)}{1+y(t-2)^{9.65}} - y(t). \tag{4.4}$$

В качестве интервала интегрирования выберем $[0, 100]$, а в качестве функции истории $y(t) = 0.5$ при $t \leqslant 0$.

Численная процедура dde23 позволяет вычислить приближенное решение $S(t)$ на всем промежутке tspan; при этом вся полученная информация сохраняется в структуре-решении sol. После этого функция deval может быть использована для получения значений решения в точках, заданных пользователем в массиве t

```
S = deval(sol,t);
```

При такой форме вывода полученных результатов вы можете решить интересующее вас ДУЗА лишь один раз и впоследствии многократно использовать полученную структуру-решение для проведения дополнительных исследований. Численное решение само по себе является непрерывным и для вывода его графика в виде гладкой кривой оно может быть вычислено в достаточно большом количестве точек с использованием функции deval.

В указанной работе получен график зависимости $y(t-2)$ от $y(t)$. Подобный вывод часто используется в работах по нелинейной динамике, но мы не можем получить этот график, непосредственно применяя программный код, использованный нами в примере 4.4.1. Это обусловлено тем обстоятельством, что значения, записанные в массив sol.x, не являются равноразнесенными: если точка $\hat{t}$ является элементом sol.x, в массиве sol.y имеется соответствующая аппроксимация решения $y(\hat{t})$, но в общем случае может не найтись элемента массива sol.x, соответствующего значению $\hat{t} - 2$ и, следовательно, в этом случае мы не будем располагать аппроксимацией решения в этой точке $y(\hat{t} - 2)$. Эта проблема может быть легко решена с использованием функции deval. В программе ch4ex2.m определяется массив t, элементами которого являются 1000 равноразнесенных точек

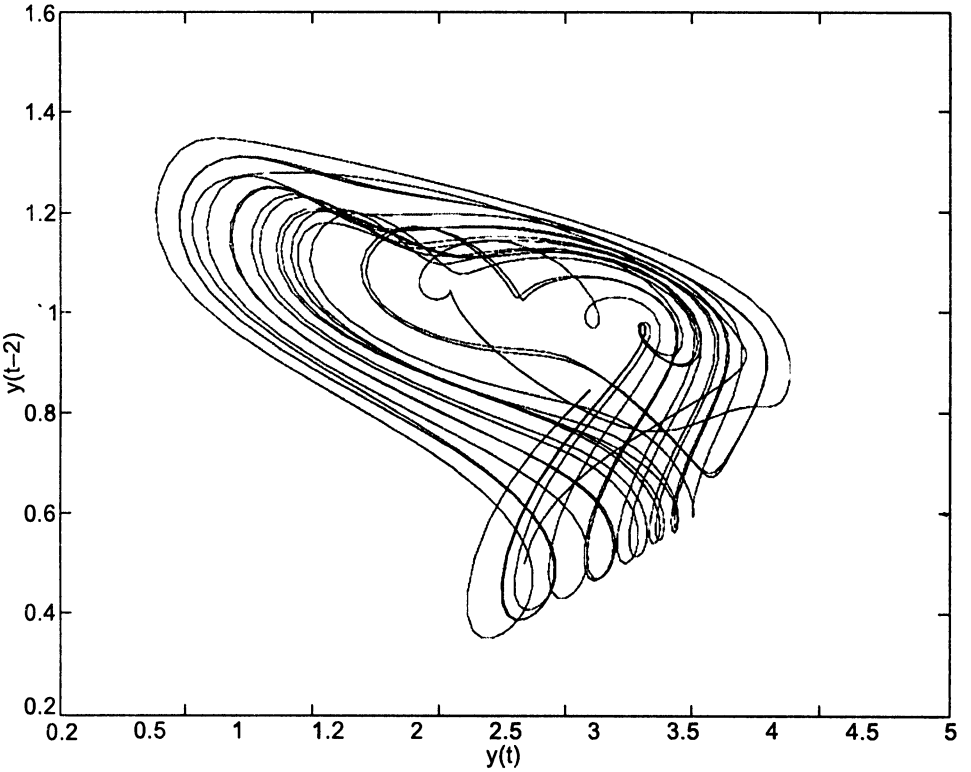

Рис. 4.2. Пример 5 из работы [Willé & Baker, 1992].

на интервале $[2, 100]$; значения решения в этих точках вычисляются с использованием функции `deval`. После этого можно повторно использовать функцию `deval` для вычисления значений решения для элементов массива `t`, равных $t - 2$. Используя этот подход, мы получаем значения аппроксимаций для функций $y(t)$ и $y(t-2)$ для одних и тех же значений t. Может показаться, что в результате будет получено слишком много точек, но в функции `deval` выполняется лишь вычисление кусочно-кубической полиномиальной аппроксимации и в реализации этой процедуры используются быстро выполняющиеся внутренние функции MATLAB, а также векторизация вычислений, что обусловливает малую вычислительную трудоемкость всей вышеописанной операции и позволяет легко получить гладкий график решения, изображенный на рисунке 4.2.

Полный текст программы вычисления и графического вывода зависимости $y(t-2)$ от $y(t)$ приведен ниже.

```
function sol = ch4ex2

sol = dde23(@ddes,2,0.5,[0, 100]);

t = linspace(2,100,1000);
y = \ltext{deval}(sol,t);
```

```
ylag = \ltext{deval}(sol,t - 2);
plot(y,ylag)
xlabel('y(t)');
ylabel('y(t - 2)');

%================================
function dydt = ddes(t,y,Z)
dydt = 2*Z/(1 + Z^9.65) - y;
```

ПРИМЕР 4.4.3

Рассмотрим модель долгосрочного партнерства в однородной популяции, использованную в работе [Corwin, Sarafyan & Thompson, 1997] для исследования процессов передачи ВИЧ-ифекции. Построение этой модели в виде системы ОДУ и ДУЗА является сложной проблемой, но после получения этой системы уравнений моделирование происходящих в популяции процессов может быть легко выполнено с использованием численной процедуры dde23. В исследуемой здесь математической модели переменная x соответствует числу восприимчивых к болезни особей, y — числу инфицированных особей, λ — коэффициент, определяющий скорость перехода особей популяции из состояния восприимчивых к болезни в состояние инфицированных (за единицу времени), D — длительность периода долгосрочного партнерства, G — коэффициент, определяющий скорость перехода особей популяции из инфицированного состояния в состояние восприимчивых к болезни, c — число сексуальных контактов во время долгосрочного партнерства, ведущих к инфицированию (для каждой персоны за единицу времени), n — первоначальная численность популяции. Тогда для $t \leqslant D$ система уравнений имеет следующий вид

$$\begin{aligned} x'(t) &= -x(t)\lambda(t) + Gy(t), \\ n &= x(t) + y(t), \\ \lambda(t) &= \frac{c}{n}\left\{\int_0^t \int^t e^{-G(t-w)}x(w)\lambda(w)dwds + \int_0^t e^{-G(t-s)}y(s)ds\right\}, \end{aligned}$$

и для $t \geqslant D$, соответственно,

$$\begin{aligned} x'(t) &= -x(t)\lambda(t) + Gy(t), \\ n &= x(t) + y(t), \\ \lambda(t) &= \frac{c}{n}\left\{\int_{t-D}^t \int^t e^{-G(t-w)}x(w)\lambda(w)dwds + \int_{t-D}^t e^{-G(t-s)}y(s)ds\right\}. \end{aligned}$$

Эти системы принадлежат к классу алгебро-дифференциальных уравнений Вольтерра. В некоторых случаях подобные системы уравнений могут быть численно решены с использованием методов, предназначенных для решения систем ДУЗА

и рассматриваемый пример относится именно к этой категории задач. Действительно, при $t \leqslant D$ следует решить систему ОДУ

$$\begin{aligned}
x'(t) &= -x(t)\lambda(t) + Gy(t),\\
y'(t) &= -x'(t),\\
\lambda'(t) &= (c/n)e^{-Gt}\{(I_1'(t) + I_2'(t)) - G(I_1(t) + I_2(t))\},\\
I_1'(t) &= e^{Gt}y(t),\\
I_2'(t) &= te^{Gt}x(t)\lambda(t),\\
I_3'(t) &= e^{Gt}x(t)\lambda(t),
\end{aligned}$$

а при $t \geqslant D$, соответственно, систему ДУЗА

$$\begin{aligned}
x'(t) &= -x(t)\lambda(t) + Gy(t),\\
y'(t) &= -x'(t),\\
\lambda'(t) &= (c/n)e^{-Gt}\{(I_1'(t) + I_2'(t)) - G(I_1(t) + I_2(t))\},\\
I_1'(t) &= e^{Gt}y(t) - e^{G(t-D)}y(t-D),\\
I_2'(t) &= De^{Gt}x(t)\lambda(t) - I_3(t),\\
I_3'(t) &= e^{Gt}x(t)\lambda(t) - e^{G(t-D)}x(t-D)\lambda(t-D).
\end{aligned}$$

Отметим, что нет необходимости во внесении в рассматриваемую систему уравнений для компоненты решения $y(t) = n - x(t)$, однако на практике подобное представление математической модели удобно тем, что позволяет получить значения этой компоненты одновременно со значениями $x(t)$ и $\lambda(t)$. Вычислим интегралы в выражениях для $\lambda(t)$. Рассмотрим более сложный случай $t \geqslant D$. Запишем функцию $\lambda(t)$ в виде

$$\lambda(t) = \frac{c}{n}e^{-Gt}(I_1(t) + I_2(t)), \tag{4.5}$$

где

$$\begin{aligned}
I_1(t) &= \int_{t-D}^{t} e^{Gs}y(s)ds,\\
I_2(t) &= \int_{t-D}^{t}\int^{t} e^{Gw}x(w)\lambda(w)dwds.
\end{aligned}$$

Дифференцируя выражение (4.5), получаем приведенное выше ОДУ для функции $\lambda(t)$. Таким образом, во время решения задачи необходимо лишь вычислять производные этих интегралов. С использованием правила Лейбница выражение для производной первого интеграла может быть переписано в следующей форме

$$I_1'(t) = e^{Gt}y(t) - e^{G(t-D)}y(t-D).$$

Аналогично для производной второго интеграла имеем

$$I_2'(t) = \int_{t-D}^{t} e^{Gt}x(t)\lambda(t)ds - \int_{t-D}^{t} e^{Gs}x(s)\lambda(s)ds.$$

Вычисление первого интеграла в правой части этого выражения не вызывает труда. Если определить $I_3(t) = \int_{t-D}^{t} e^{Gs}x(s)\lambda(s)ds$, то для вычисления значений этой новой переменной следует продифференцировать это равенство и численно решить полученное ДУЗА

$$I_3'(t) = e^{Gt}x(t)\lambda(t) - e^{G(t-D)}x(t-D)\lambda(t-D)$$

совместно с основной системой. Аналогично, определив

$$I_2'(t) = De^{Gt}x(t)\lambda(t) - I_3(t),$$

можно получить соответствующее ДУЗА для вычисления соответсвующей переменной.

Вывод уравнений при $t \leqslant D$ выполняется аналогичным образом, но в этом случае нет необходимости в вычислении интеграла $I_1(t)$. Значение этого интеграла потребуется нам лишь в момент $t = D$, когда наша программа переключается на решение системы ДУЗА. По определению все интегралы равны нулю в момент $t = 0$, что определяет начальные значения для соответствующих дифференциальных уравнений. При сравнении определений интегралов для двух участков временно́й оси можно заметить, что эти интегралы являются непрерывными функциями в $t = D$ и соответствующие значения, полученные на первом участке временно́й оси, могут быть использованы в качестве начальных значений для соответствующих ДУЗА для случая $t \geqslant D$. Для удобства вычисления $I_1(t)$ на всем интервале времени следует доопределить соответствующее дифференциальное уравнение для его использования в случае $t \leqslant D$. Это должно быть сделано так, чтобы значение решения этого уравнения в момент $t = D$ было равно требуемому начальному значению для рассмотренного выше дифференциального уравнения, определенного на втором участке временно́й оси.

Рассматриваемая в этом примере задача решается при значениях констант

$$c = 0.5,\ n = 100,\ G = 1,\ D = 5$$

и при начальных данных

$$x(0) = 0.8n,\ y(0) = 0.2n,\ \lambda(0) = 0.$$

При написании программного кода функций для вычисления уравнений мы должны переписать уравнение для $y'(t)$ так, чтобы производных в правой части этого уравнения не было. Очевидно, что для этого следует использовать уравнение для $x'(t)$. В принципе, это должно было быть сделано в самом начале нашего исследования, но представленная выше немодифицированная система уравнений позволяет более наглядно раскрыть особенности модели и детали ее вывода. Кроме того, это позволило проиллюстрировать полезную с практической точки зрения методику написания программного кода для вычисления уравнений: вместо исключения $x'(t)$ из уравнения для $y'(t)$ можно вычислить значение $x'(t)$ и использовать его для вычисления $y'(t)$. Аналогично, предварительно вычисленные значения $I_1'(t)$ и $I_2'(t)$ могут быть использованы для вычисления $\lambda'(t)$.

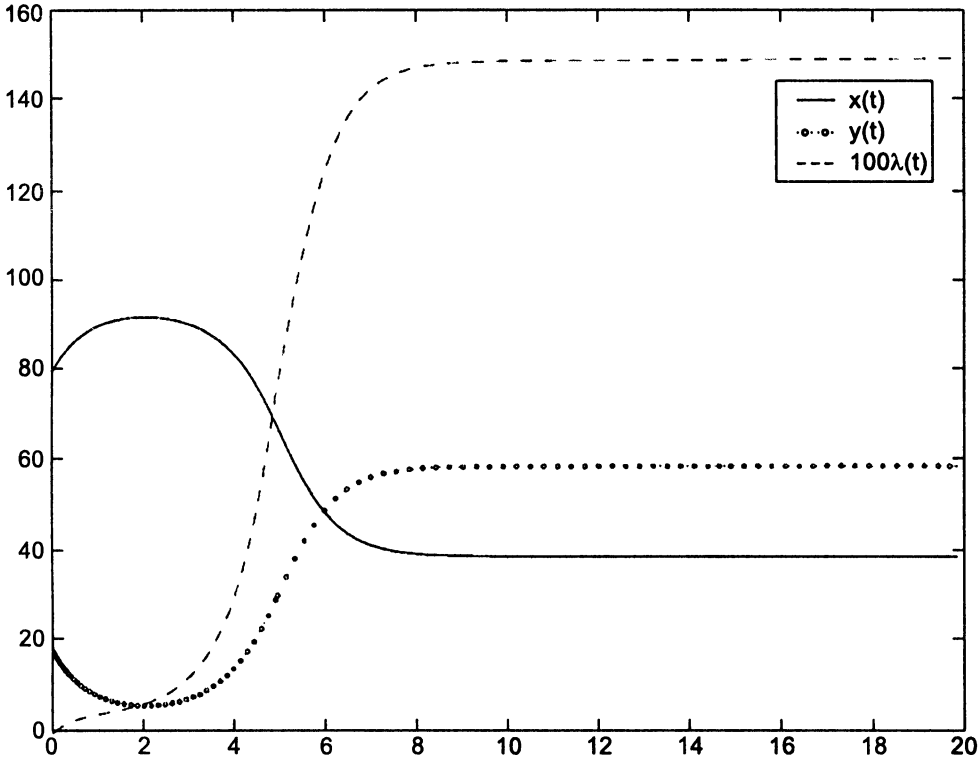

Рис. 4.3. Модель долгосрочного партнерства и процесса передачи ВИЧ-инфекции.

Реализация численной процедуры `dde23` не предполагает, что функции $y(t-\tau_j)$ в обязательном порядке должны входить в решаемые уравнения. Поэтому эту процедуру можно использовать для решения ОДУ. Тем не менее, поскольку предполагается, что эта процедура предназначена для решения ДУЗА, в список входных аргументов функции, определяющей систему уравнений, необходимо включить `z`. Если с использованием этой процедуры решается система ОДУ, в качестве входного массива запаздываний следует использовать пустой массив, в противном случае численная процедура будет пытаться решить потенциальную проблему нарушения непрерывности вследствие наличия в системе запаздываний и ограничит шаг интегрирования несмотря на то, что в подобной ситуации этой проблемы не возникает. В программе `ch4ex3.m` сначала выполняется решение системы ОДУ при $t \leqslant D$ и после этого выполняется решение системы ДУЗА при $t \geqslant D$. В этом примере показано, как структура-решение, полученная на первом этапе интегрирования, может быть непосредственно использована в качестве истории для последующего этапа интегрирования (см. обсуждение этой методики в параграфе 4.3). Кроме того, после очередного этапа интегрирования структура-решение агрегируется и образует единую структуру-решение для просчитанного интервала времени. Результатами работы программы являются графики переменных x, y и 100λ, изображенные на рисунке 4.3.

```
function sol = ch4ex3
global c D G n
c = 0.5; D = 5; G = 1; n = 100;

% x = y(1), y = y(2), lambda = y(3),
% I_1 = y(4), I_2 = y(5), I_3 = y(6).
y0 = [0.8*n; 0.2*n; 0; 0; 0; 0];

sol = dde23(@odes,[],y0,[0, D]);
sol = dde23(@ddes,D,sol,[D,4*D]);

plot(sol.x,[sol.y(1:2,:); 100*sol.y(3,:)]);
legend('x(t)','y(t)','100\lambda(t)',0)

%============================================================
function dydt = odes(t,y,Z)
global c D G n
dydt = zeros(6,1);
dydt(1) = - y(1)*y(3) + G*y(2);
dydt(2) = - dydt(1);
dydt(4) = exp(G*t)*y(2);
dydt(5) = t*exp(G*t)*y(1)*y(3);
dydt(6) = exp(G*t)*y(1)*y(3);
dydt(3) = (c/n)*exp(-G*t)*((dydt(4)+dydt(5))-G*(y(4)+y(5)));

function dydt = ddes(t,y,Z)
global c D G n
dydt = zeros(6,1);
dydt(1) = - y(1)*y(3) + G*y(2);
dydt(2) = - dydt(1);
dydt(4) = exp(G*t)*y(2) - exp(G*(t - D))*Z(2);
dydt(5) = D*exp(G*t)*y(1)*y(3) - y(6);
dydt(6) = exp(G*t)*y(1)*y(3) - exp(G*(t - D))*Z(1)*Z(3);
dydt(3) = (c/n)*exp(-G*t)*((dydt(4)+dydt(5))-G*(y(4)+y(5)));
```

На начальных этапах исследования этой задачи применялись численные процедуры решения ЗНУ. При этом исследуемая система определялась для переменных $x(t)$, $y(t)$ и $\lambda(t)$. Для приближенного вычисления интегралов в выражении для коэффициента $\lambda(t)$ использовались квадратурные формулы, но в этом случае ситуация осложняется тем обстоятельством, что подынтегральные выражения зависят от x, y и самого коэффициента λ. Вследствие этого, первая версия программы, используемая нами для решения этой задачи, выполнялась на мощном центральном компьютере несколько часов и при этом часто случались прерывания вычислительного процесса. Были выполнены дополнительные аналитические исследования и усовершенствование алгоритмов применяемых численных методов, что позволило уменьшить время выполнения программы до нескольких секунд на обычном персональном компьютере. Другие важные аспекты этой задачи рассмотрены в упражнении 4.5.

ПРИМЕР 4.4.4

Модель четырехлетнего цикла развития популяции леммингов рассматривается в работе [Tavernini, 1996]. Соответствующее уравнение

$$y'(t) = ry(t)\left(1 - \frac{y(t-0.74)}{m}\right) \tag{4.6}$$

решается на интервале $[0, 40]$. Параметры имеют значения $r = 3.5$ и $m = 19$. Заметим, что при этих значениях параметров уравнение имеет постоянное (установившееся) решение $y(t) = m = 19$. В указанной работе это решение использовалось в качестве функции истории и исследовалось влияние возмущений начальных значений, отклоняющих решение от установившегося режима. В рассматриваемом здесь примере подобным начальным значением является $y(0) = 19.001$. Если начальное значение отличается от соответствующего значения функции истории, оно должно задаваться с использованием опции `InitialY`. Разрыв в начальной точке настолько мал, что возникает необходимость в задании более жестких требований на допустимые ошибки вычислений. Эти значения задаются с использованием функции `ddeset` аналогично тому, как это делается в случае решения ЗНУ с использованием функции `odeset`.

Поскольку решение демонстрирует периодическое поведение, решение полезно выводить в виде зависимости производной $y'(t)$ от значения решения $y(t)$, т. е. на фазовой плоскости. Это может быть легко сделано, т. к. в структуре-решении имеется не только поле `sol.y`, в котором содержится аппроксимация решения в узлах сетки, но и поле `sol.yp`, содержащее аппроксимацию $y'(t)$. На рисунке 4.4 показан фазовый портрет системы, полученный с использованием программы `ch4ex4.m`.

Наибольший интерес вызывают минимальные и максимальные значения численности популяции. Для мониторинга достижения этих точек может использоваться аппарат локализации событий аналогично тому, как это делается в случае использования численных процедур решения ЗНУ. Заметим, что локальные минимумы достигаются при $y'(t) = 0$, но это равенство выполнено и для точек максимума. Эти два вида событий различаются тем, что в случае локального минимума производная $y'(t)$ возрастает после прохождения этой точки, а в случае локального максимума — убывает. Аргумент `direction` функции обработки событий позволяет учесть это различие. Информация о событиях возвращается численной процедурой `dde23` в виде соответствующих полей структуры-решения, а не в виде выходных аргументов, как в случае решения стандартной ЗНУ с использованием численной процедуры `ode23`. В программе `ch4ex4.m` функция `find` применяется для определения индексов элементов массива выходных данных, соответствующих указанным двум событиям. С использованием этих индексов можно извлечь информацию, связанную с интересующим нас событием. В частности, на графике решения $y(t)$ (см. рис. 4.5) точки минимума помечены квадратиками, а точки максимума — кружочками.

Отметим, что константы r и m выступают в качестве параметров и их значения переданы в численную процедуру `dde23` посредством их перечисления в списке опциональных аргументов. Так же как и в случае использования численных процедур

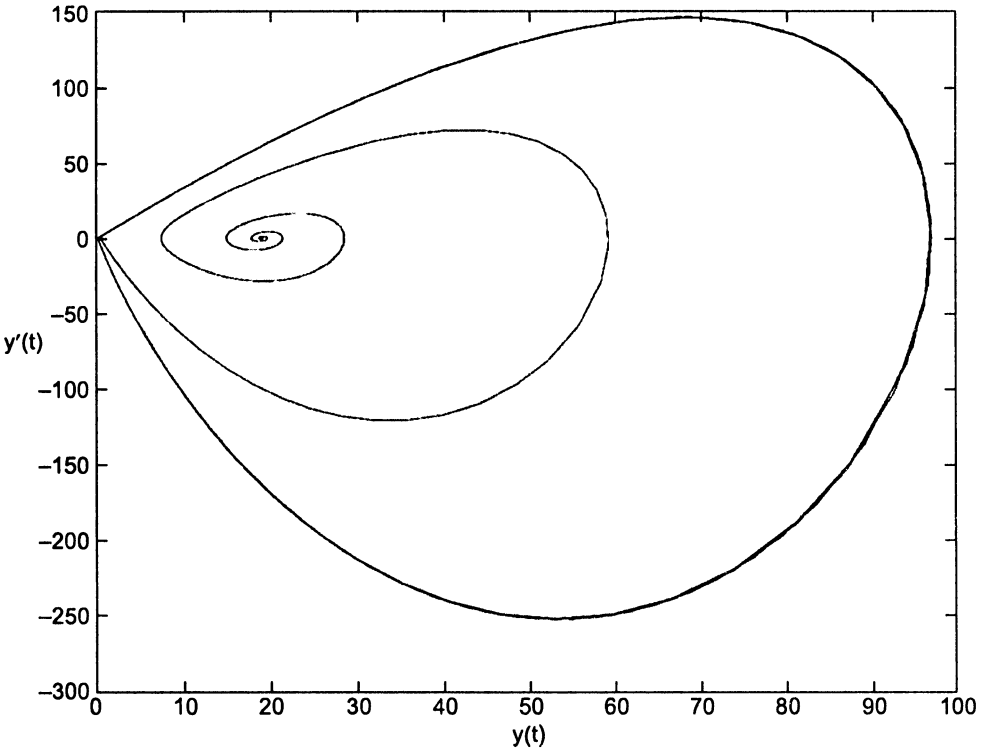

Рис. 4.4. Численность популяции леммингов — фазовый портрет.

решения ЗНУ, эти параметры должны появиться в списке аргументов функции, в которой вычисляется исследуемая система ДУЗА, а также в списке аргументов функции обработки событий, даже если эти параметры не используются в этих функциях.

```
function sol = ch4ex4
r = 3.5; m = 19;
options = ddeset('Events',@events,'InitialY',19.001,...
                 'RelTol',1e-4,'AbsTol',1e-7);
sol = dde23(@ddes,0.74,19,[0, 40],options,r,m);
plot(sol.y,sol.yp);
xlabel('y(t)');
ylabel('y''(t)');

n1 = find(sol.ie == 1);
x1 = sol.xe(n1);
y1 = sol.ye(1,n1);
n2 = find(sol.ie == 2);
x2 = sol.xe(n2);
y2 = sol.ye(1,n2);
```

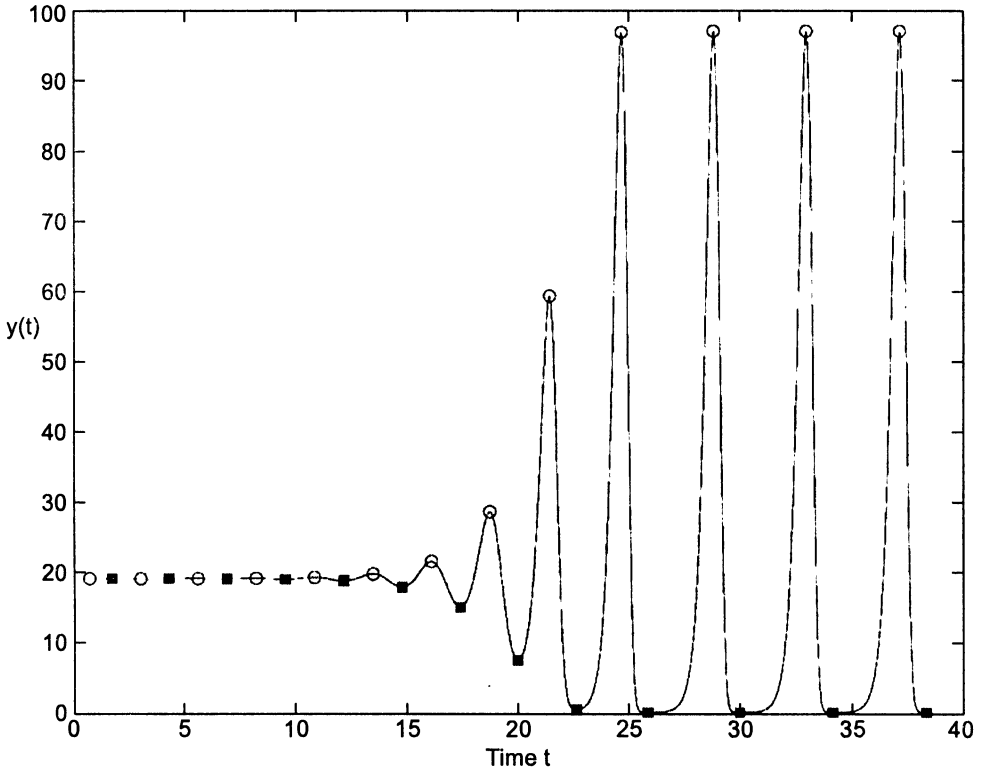

Рис. 4.5. Численность популяции леммингов — зависимость от времени.

```
figure
plot(sol.x,sol.y,'k',x1,y1,'rs',x2,y2,'bo')
xlabel('Time t');
ylabel('y(t)');

%=======================================================
function dydt = ddes(t,y,Z,r,m)
dydt = r*y*(1 - Z/m);

function [value,isterminal,direction] = events(t,y,Z,r,m)
dydt = ddes(t,y,Z,r,m);
value = [dydt; dydt];
direction = [+1; -1];
isterminal = [0; 0];
```

ПРИМЕР 4.4.5

Выше рассматривались примеры и упражнения, в которых нарушения непрерывности, имеющие место в заранее известные моменты времени, обрабатывались с использованием опций `Jumps` и `InitialY`. Нарушения непрерывности других типов зависят от самого решения и поэтому соответствующие события должны быть

локализованы с использованием опции Events. Если в результате возникновения события происходит замена уравнений, процедура численного интегрирования должна быть запущена заново. Ранее был рассмотрен пример, когда этот перезапуск численной процедуры происходил в заранее известный момент времени. Однако в случаях, когда события зависят от решения, нам не известно в какой точке и сколько раз произойдут эти события и, следовательно, мы не можем заранее сказать, в каких точках произойдут новые запуски численных процедур и сколько потребуется подобных запусков. В этом параграфе мы рассмотрим пример, соответствующий подобной ситуации. Постановка такой задачи в виде, допускающем такую формализацию, что оказывается возможным реализовать понятный и удобный пользовательский интерфейс численной процедуры решения ДУЗА, является непростой проблемой. Другой пример, в котором используется опция Events, рассматривается в упражнении 4.14.

Двухколесный дорожный чемодан может начать колебаться из стороны в сторону при перевозке. При этом человек, тянущий этот чемодан за собой, пытается вернуть его в номинальное (вертикальное) положение, прикладывая к ручке чемодана восстанавливающий угловой момент. Разумеется, существует некоторое запаздывание между моментом отклонения чемодана от вертикального положения и моментом реакции человека, величина которого существенно влияет на устойчивость движения. Этот процесс был промоделирован в работе [Suherman et al., 1997] с использованием ДУЗА

$$\theta''(t) + \mathrm{sign}(\theta(t))\gamma\cos(\theta(t)) - \sin(\theta(t)) + \beta\theta(t-\tau) = A\sin(\Omega t + \eta),$$

где $\theta(t)$ — угол отклонения чемодана от вертикального положения. Это уравнение решается на интервале интегрирования $[0, 12]$ после предварительного преобразования к системе двух уравнений первого порядка с переменными $y_1(t) = \theta(t)$ и $y_2(t) = \theta'(t)$. На рисунке 3 из указанной работы показан график компоненты решения $y_1(t)$, а также фазовый портрет траектории $(y_1(t), y_2(t))$, полученные при значениях констант

$$\gamma = 2.48,\ \beta = 1,\ \tau = 0.1,\ A = 0.75,\ \Omega = 1.37,\ \eta = \arcsin(\gamma/A)$$

и нулевой векторной функцией истории. Колесо касается земли (т. е. чемодан принимает вертикальное положение) при $y_1(t) = 0$. После этого события процесс интегрирования должен быть запущен заново при $y_1(t) = 0$ и при новом значении компоненты $y_2(t)$, полученном путем умножения этой величины на коэффициент упругого восстановления, равный в рассматриваемом примере 0.913. Чемодан считается опрокинувшимся, если $|y_1(t)| = \pi/2$ и соответствующее событие является терминальным.

Задача решается с использованием следующей программы.

```
function sol = ch4ex5a
state = +1;
opts = ddeset('RelTol',1e-5,'Events',@events);
sol = dde23(@ddes,0.1,[0; 0],[0 12],opts,state);
```

```
ref = [4.516757065, 9.751053145, 11.670393497];
fprintf('Kind of Event: dde23 reference\n');
event = 0;
while sol.x(end) < 12
    event = event + 1;
    if sol.ie(end) == 1
        fprintf('A wheel hit the ground. %10.4f %10.6f\n',...
                sol.x(end),ref(event));
        state = - state;
        opts = ddeset(opts,'InitialY',[ 0; 0.913*sol.y(2,end)]);
        sol = dde23(@ddes,0.1,sol,[sol.x(end) 12],opts,state);
    else
        fprintf('The suitcase fell over. %10.4f %10.6f\n',...
                sol.x(end),ref(event));
        break;
    end
end
plot(sol.y(1,:),sol.y(2,:))
xlabel('\theta(t)')
ylabel('\theta''(t)')

%=========================================================================
function dydt = ddes(t,y,Z,state)
gamma = 0.248; beta = 1; A = 0.75; omega = 1.37;
ylag = Z(1,1); dydt = [y(2); 0];
dydt(2) = sin(y(1)) - state*gamma*cos(y(1)) - beta*ylag ...
          + A*sin(omega*t + asin(gamma/A));

function [value,isterminal,direction] = events(t,y,Z,state)
value = [y(1); abs(y(1))-pi/2];
isterminal = [1; 1]; direction = [-state; 0];
```

С использованием этой программы был получен фазовый портрет системы, изображенный на рисунке 4.6 и соответствующий результату, представленному в вышеуказанной работе. Кроме того, программа вывела информацию о локализованных событиях

```
>> ch4ex5a;
Kind of Event:                dde23    reference
A wheel hit the ground.      4.5168     4.516757
A wheel hit the ground.      9.7511     9.751053
The suitcase fell over.     11.6704    11.670393
```

Референсные значения (reference) были вычислены с использованием программы DKLAG5 из вышеуказанной работы [Suherman et al., 1997] и проверены с использованием усовершенствованной версии этой программы DKLAG6, кратко описанной в подпараграфе 4.5.

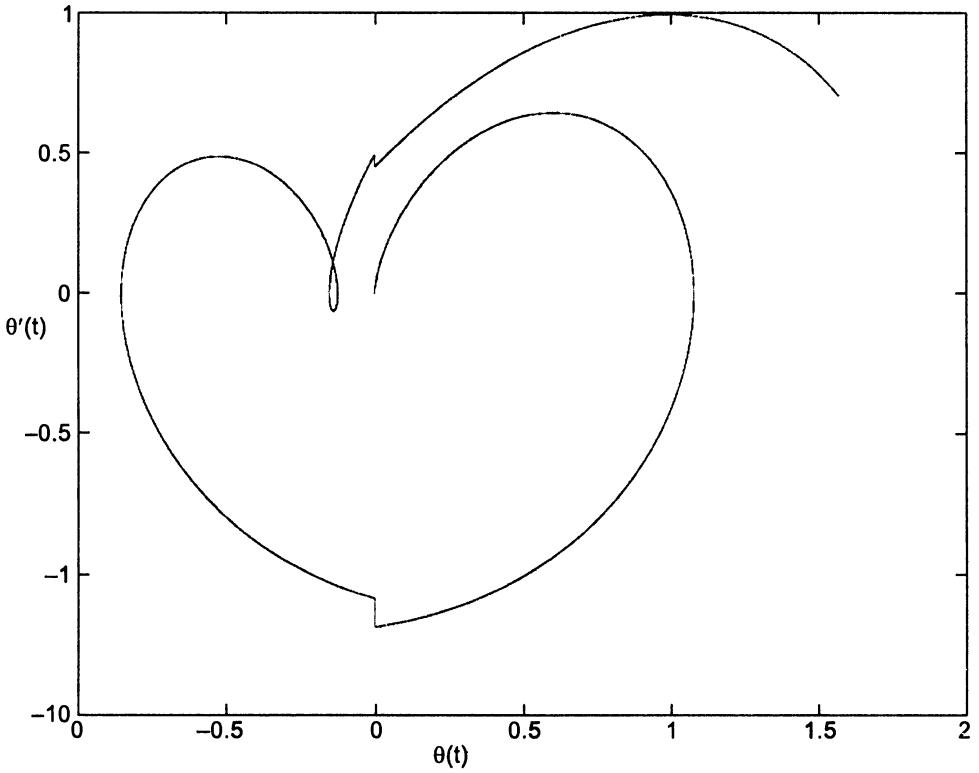

Рис. 4.6. Задача о движении двухколесного дорожного чемодана.

Рассматриваемая модель непроста и поэтому некоторые аспекты ее программной реализации будут рассмотрены ниже более детально. Написание программного кода функции, реализующей вычисление исследуемого ДУЗА, не составляет труда, но следует внимательно подойти к проблеме вычисления коэффициента $\mathrm{sign}(y_1(t))$, имеющего разрыв. Для этого необходимо назначить параметру `state` значение $+1$ и изменять его знак при каждом выходе из процедуры `dde23`, т. к. в этом случае $y_1(t)$ обращается в ноль. Указанный способ изменения значения параметра `state` гарантирует, что в численной процедуре `dde23` не будет происходить обработка ситуаций, связанных с нарушениями непрерывности: если вычисление производной реализовано так, что параметр `state` изменяется перед новым запуском численной процедуры интегрирования системы, обработка нарушений непрерывности будет происходить (эта проблема обсуждается в работе [Shampine & Thompson, 2000]). После вызова численной процедуры `dde23` необходимо проанализировать ситуацию, при которой произошло прекращение ее работы. Одним из возможных вариантов является то, что был достигнут конец интервала интегрирования: в этом случае значение `sol.x(end)` должно быть равно 12. Другой возможной причиной прекращения вычислений является то обстоятельство, что чемодан опрокинулся: в этом случае значение `sol.ie` должно быть равно 2. Обе

эти ситуации являются терминальными и работа программы прекращается. Более интересной является ситуация, когда колесо коснулось земли, т.е. $y_1(t) = 0$: в этом случае `sol.ie` должно быть равно 1. После этого знак параметра `state` изменяется и процесс интегрирования запускается заново. Поскольку колесо после контакта с поверхностью земли отскакивает, решение в конце текущего этапа интегрирования `sol.y(:,end)` должно быть модифицировано таким образом, чтобы оно могло использоваться в качестве начальных данных на следующем этапе интегрирования. Опция `InitialY` используется для задания начального значения в случае, если оно не совпадает с соответствующим значением функции истории. Событие $y_1(t) = 0$, приводящее к прекращению текущего этапа интегрирования, имеет место и в начальный момент на следующем этапе интегрирования. Так же как и численные процедуры решения ЗНУ, процедура `dde23` не прерывает работу в специальном случае возникновения события в начальной точке, но оно локализуется и регистрируется. Чтобы избежать этого рекомендуется соответствующим образом определить функцию обработки событий. Если значение параметра `state` равно $+1$ (соответственно, -1), необходимо выполнить локализацию соответствующего события для определения точки, где компонента решения $y_1(t)$, уменьшаясь (соответственно, увеличиваясь), обращается в ноль. Численная процедура информируется об этом событии посредством изменения значения первой компоненты аргумента `direction` на `-state`. В цикле `while` функция `ddeset` используется для изменения текущей структуры опций. Напомним, что в случае решения ЗНУ аналогичное действие может быть выполнено с использованием соответствующей функции `odeset`. В заключительной части программы содержатся операторы, обеспечивающие вывод результатов вычислений. Решения с приемлемой точностью могут быть получены при значениях допустимых ошибок, принятых по умолчанию, но для обеспечения гладкости выводимой траектории решения в фазовой плоскости были вычислены дополнительные точки. При увеличении допустимого значения относительной ошибки до $1e-5$ оказалось возможным обеспечить более точное соответствие референсным значениям.

Длительность процесса вычислений оказалась небольшой и поэтому можно выполнить некоторые дополнительные вычисления, позволяющие выявить интересные особенности движения исследуемой системы. При фиксированном наборе остальных параметров существует некоторое критическое значение амплитуды A_{cr}, равное минимальному значению A, при котором чемодан опрокидывается. В работе [Suherman et al., 1997] приведены графики зависимости величины A_{cr} от частоты возбуждения Ω для нескольких наборов параметров задачи. Например, при $\beta = 1$, $\tau = 0.1$ и $\Omega = 2$ значение критической амплитуды находится между 0.4 и 1.4. Величину A_{cr} можно вычислить следующим образом. Определим функцию $f(A)$, которая принимает значение $+1$, если чемодан не опрокинулся в процессе интегрирования и значение -1, если чемодан опрокинулся. Тогда с использованием метода деления пополам находится точка, в которой функция $f(A)$ изменяет знак. При использовании подобного подхода предполагается, что чемодан не опрокидывается при $A < A_{cr}$ и опрокидывается при бóльших значениях амплитуды A. (Интересным аспектом этой задачи является то обстоятельство, что при некотором выборе параметров вторая часть этого предположения выполняется не всегда.) В качестве интервала интегрирования использовался $[0, \frac{40\pi}{\Omega}]$,

что соответствует двадцати циклам воздействия момента возбуждения системы. В версии программы `ch4ex5a.m` (с вычислением указанной функции) операторы вывода результатов вычислений удалены; амплитуда возбуждения A передается в процедуру как дополнительный параметр. Изменение знака выявляется с использованием функции `fzero`. Для уменьшения времени выполнения программы мы использовали в функциях `fzero` и `dde23` умеренные значения допустимых ошибок вычислений. Для каждого вычисления функции $f(A)$ с использованием `fzero` требуется решать рассматриваемое ДУЗА. Для индикации продолжения работы программы в условиях, когда основной вывод программы запрещен, мы использовали трассировку выполнения функции `fzero`. Вычисленное значение критической амплитуды оказалось приблизительно равным $A_{cr} = 0.93$. Листинг программы приведен ниже.

```
function ch4ex5b
options = optimset('Display','iter','TolX',0.01);
Acr = fzero(@f,[0.4, 1.4],options);
fprintf('\nThe critical excitation amplitude is %4.2f.\n',Acr);

%================================================
function fval = f(A)
fval = +1;
omega = 2;
tfinal = 40*pi/omega;
state = +1;
opts = ddeset('Events',@events);
sol = dde23(@ddes,0.1,[0; 0],[0 tfinal],opts,state,A);
while sol.x(end) < tfinal
   if sol.ie(end) == 1
      state = - state;
      opts = ddeset(opts,'InitialY',[ 0; 0.913*sol.y(2,end)]);
      sol = dde23(@ddes,0.1,sol,[sol.x(end) tfinal],opts,state,A);
   else
      fval = -1;
      break;
   end
end

function dydt = ddes(t,y,Z,state,A)
omega = 2; gamma = 0.248; beta = 1;
ylag = Z(1,1);
dydt = [y(2); 0];
dydt(2) = sin(y(1)) - state*gamma*cos(y(1)) - beta*ylag ...
          + A*sin(omega*t + asin(gamma/A));

function [value,isterminal,direction] = events(t,y,Z,state,A)
value = [y(1); abs(y(1))-pi/2];
isterminal = [1; 1];
direction = [-state; 0];
```

■ УПРАЖНЕНИЕ 4.2

В эпидемической модели Кука, рассмотренной в работе [MacDonald, 1978], для описания динамики численности инфицированной части популяции $y(t)$ используется следующее уравнение

$$y'(t) = by(t-7)[1-y(t)] - cy(t),$$

где b и c — положительные константы. Это уравнение решается на интервале $[0, 60]$ с функцией истории $y(t) = \alpha$ при $t \leqslant 0$. Константа α удовлетворяет $0 < \alpha < 1$.

Напишите процедуру с входными аргументами α, b и c, предназначенную для получения решения этого ДУЗА с использованием `dde23`, и выведите с ее помощью график решения. В заголовке графика должны быть указаны используемые при вычислениях значения α, b и c. Очевидно, что при любых b и c точка $y(t) = 0$ является точкой равновесия (стационарным решением). При $b > c$ существует вторая точка равновесия $y(t) = 1 - c/b$. Выполнив несколько численных экспериментов с различными значениями α, b и c, убедитесь, что при $b > c$ решения стремятся ко второй точке равновесия, в противном случае — к первой. Вычислите решение при значениях констант $\alpha = 0.8$, $b = 2$ и $c = 1$ и выведите его график. Убедитесь, что это решение стремится к нетривиальной точке равновесия.

■ УПРАЖНЕНИЕ 4.3

В работе [Neves, 1975] рассмотрена задача с функцией истории, не являющейся постоянной. Исследуемая система ДУЗА

$$\begin{aligned}
y_1'(t) &= y_5(t-1) + y_3(t-1),\\
y_2'(t) &= y_1(t-1) + y_2(t-0.5),\\
y_3'(t) &= y_3(t-1) + y_1(t-0.5),\\
y_4'(t) &= y_5(t-1)y_4(t-1),\\
y_5'(t) &= y_1(t-1)
\end{aligned}$$

должна быть решена на интервале $0 \leqslant t \leqslant 1$ с функцией истории, определяемой при $t \leqslant 0$ следующими равенствами

$$\begin{aligned}
y_1(t) &= e^{t+1},\\
y_2(t) &= e^{t+05},\\
y_3(t) &= \sin(t+1),\\
y_4(t) &= y_1(t),\\
y_5(t) &= y_1(t).
\end{aligned}$$

Решите эту задачу и выведите графики всех компонент решения. Программа, которую вам следует написать, будет аналогична `ch4ex1.m`, но вычисление функции истории должно быть реализовано в виде отдельной процедуры, и указатель на эту процедуру должен выступать в качестве соответствующего аргумента численной процедуры `dde23`. Напомним, что выходными аргументами процедуры для вычисления исследуемой системы ДУЗА и функции истории являются вектор-столбцы.

■ УПРАЖНЕНИЕ 4.4

В работе [Farmer, 1982] получены графики сечений Пуанкаре для уравнения Маккея–Гласса (Mackey–Glass), которое демонстрирует хаотическое поведение

$$y'(t) = \frac{0.2y(t-14)}{1+y(t-14)^{10}-0.1y(t)}.$$

Решите это скалярное ДУЗА на интервале $[0,300]$ с функцией истории $y(t) = 0.5$ при $t \leqslant 0$. Выведите график зависимости $y(t-14)$ от $y(t)$. Сравните полученный результат с рисунком 2а из вышеуказанной работы. Во избежание загромождения графика деталями переходного процесса, графический вывод результатов начните с момента $t = 50$. Для того, чтобы точно воспроизвести рисунок 2а из вышеуказанной работы, сформируйте массив из 1000 равноразнесенных на интервале $[50,300]$ точек, вычислите значения $y(t)$ в этих точках, а затем вычислите значения функции $y(t-14)$. Написанная вами программа должна быть во многом аналогична программе `ch4ex2.m`.

■ УПРАЖНЕНИЕ 4.5

Решение задачи о долгосрочном партнерстве и распространении ВИЧ-инфекции, полученное в примере 4.4.3 с использованием программы `ch4ex3.m`, стремится к некоторому стационарному решению x_s, y_s, λ_s. Покажите, что существует два стационарных решения: тривиальное

$$\lambda_s = 0,\ x_s = n,\ y_s = 0$$

и нетривиальное, представляющее особый интерес

$$\lambda_s = cD - G,\ x_s = \frac{Gn}{\lambda_s + G},\ y_s = n - x_s.$$

Для этого вернитесь к рассмотрению интегральной формулировки задачи для случая $t \geqslant D$ и предположите, что при больших t решение является постоянной функцией. Модифицируйте текст программы `ch4ex3.m` так, чтобы можно было найти решение задачи при значениях $G = 0.1$, 1 и 2 и убедитесь в том, что предельные значения `sol.y(1:3,end)` достаточно точно совпадают с аналитически вычисленным стационарным решением. В этой модели константа G определяет скорость перехода особей популяции из инфицированного состояния в состояние выздоровевших, но оставшихся восприимчивыми к болезни.

■ УПРАЖНЕНИЕ 4.6

В руководстве пользователя [Paul, 1995] для написанной на языке Fortran 77 программы `ARCHI` (более детальное обсуждение этой программы представлено в параграфе 4.5), приведен простой пример решения системы ДУЗА

$$y_1'(t) = y_1(t-1)y_2(t-2),$$
$$y_2'(t) = -y_1(t)y_2(t-2)$$

на интервале $[0, 4]$ с функцией истории $y_1(t) = \cos(t)$ и $y_2(t) = \sin(t)$ при $t < 0$ и начальными значениями $y_1(0) = 0$ и $y_2(0) = 0$. Заметим, что $y_1(t)$ терпит разрыв в начальной точке и поэтому следует использовать опцию `InitialY`. Вычислите решение и выведите его график. В примере из указанного руководства используется численная процедура с критерием контроля ошибки вычислений, обеспечивающим соблюдение лишь допустимого значения абсолютной ошибки 10^{-9}. Численная процедура `dde23` не позволяет осуществлять контроль ошибки на основании оценки лишь абсолютной ошибки, но в целях практики использования опций этой процедуры выполните вычисления, используя заданное по умолчанию значение допустимой величины относительной ошибки, а в качестве допустимой величины абсолютной ошибки `AbsTol` положите значение `1e-9`.

◘ УПРАЖНЕНИЕ 4.7

В примере 4.4 из работы [Oberle & Pesch, 1981] рассматривается модель распространения инфекции, предложенная Хоппенстедтом и Уолтмананом: уравнение

$$y'(t) = \begin{cases} -ry(t)0.4(1-t), & 0 \leqslant t \leqslant 1-c, \\ -ry(t)(0.4(1-t) + 10 - e^{\mu}y(t)), & 1-c < t \leqslant 1, \\ -ry(t)(10 - e^{\mu}y(t)), & 1 < t \leqslant 2-c, \\ -re^{\mu}y(t)(y(t-1) - y(t)), & 2-c < t \end{cases}$$

решается на интервале интегрирования $[0, 10]$ с функцией истории $y(t) = 10$ при $t \leqslant 0$; $c = 1/\sqrt{2}$ и $\mu = r/10$. В указанной работе это уравнение решалось при нескольких значениях параметра r. Читателю предлагается разработать программу, в которой величина r также должна являться параметром. Решение этой задачи должно быть получено лишь при $r = 0.5$. Именно для этого случая в указанной работе приведено референсное значение $y(10) = 0.06302089869$. Различные фазы процесса распространения инфекции описываются различными уравнениями, но необходимо, чтобы решение на всем интервале интегрирования было непрерывным. Проблема заключается в том, что смена используемых в модели уравнений, определяющих величину $y'(t)$, приводит к нарушениям непрерывности производных малого порядка. Поскольку это происходит в заранее известные моменты времени, все, что необходимо сделать для успешного решения задачи — это задать эти моменты с помощью опции `Jumps`. Вы можете запрограммировать вычисление правой части рассматриваемого ДУЗА непосредственно с использованием оператора `if`. Поскольку референсное значение решения было найдено при очень жестких требованиях на величину допустимых ошибок вычислений, в вашей программе следует использовать значения относительной и абсолютной ошибок вычислений соответственно `1e-5` и `1e-8`. Выведите график функции

$$I(t) = -\frac{y'(t)}{ry(t)},$$

используя информацию, хранящуюся в структуре-решении `sol.yp` и `sol.y`.

■ УПРАЖНЕНИЕ 4.8

Уравнения иммунологической модели [Марчук, 1991], рассмотренной в работе [Hairer, Nörsett & Wanner, 1987], имеют следующий вид

$$
\begin{aligned}
y_1'(t) &= (h_1 - h_2 y_3(t)) y_1(t), \\
y_2'(t) &= \xi(y_4(t)) h_3 y_3(t - 0.5) y_1(t - 0.5) - h_5(y_2(t) - 1), \\
y_3'(t) &= h_4(y_2(t) - y_3(t)) - h_8 y_3(t) y_1(t), \\
y_4'(t) &= h_6 y_1(t) - h_7 y_4(t),
\end{aligned}
$$

где функция

$$
\xi(y_4(t)) = \begin{cases} 1, & \text{если} \quad y_4(t) \leqslant 0.1, \\ \frac{10}{9}(1 - y_4(t)), & \text{если} \quad 0.1 < y_4(t) \leqslant 1 \end{cases}
$$

является непрерывной, но имеет нарушение непрерывности первой производной при $y_4(t) = 0.1$, что в свою очередь обусловливает нарушение непрерывности производных малого порядка компоненты решения $y_2(t)$. Уравнения решаются на интервале $[0, 60]$ с функцией истории

$$
y_1(t) = \max(0, t + 10^{-6}),\ y_2(t) = 1,\ y_3(t) = 1,\ y_4(t) = 0
$$

при $t \leqslant 0$. Как было отмечено выше, первая производная компоненты решения $y_1(t)$ имеет разрыв в точке $t = -10^{-6}$, распространяющийся на интервал интегрирования. На рисунке 15.8 в работе [Hairer et al., 1987] представлены графики решений при значениях параметров

$$
h_1 = 2,\ h_2 = 0.8,\ h_3 = 10^4,\ h_4 = 0.17,\ h_5 = 0.5,\ h_7 = 0.12,\ h_8 = 8
$$

для двух значений параметра h_6: $h_6 = 10$ и $h_6 = 300$. Используя `dde23`, решите задачу при $h_6 = 300$. Для того, чтобы воспроизвести график из работы [Hairer et al., 1987], необходимо масштабировать компоненты решения

$$
10^4 y_1,\ 0.5 y_2,\ y_3,\ 10 y_4
$$

и вывести соответствующий график на координатной плоскости (`[0 60 -1 15.5]`). Массив значений масштабированного решения `yplot`, необходимый для вывода графика, может быть легко построен с использованием команды

```
yplot = sol.y;
yplot(1,:) = 1e4*yplot(1,:);
```

(Вычисление значений для других компонент решения может быть выполнено с использованием аналогичных команд). Для получения точного решения задачи на всем интервале интегрирования необходимо уменьшить значения допустимых ошибок вычислений: положите значения относительной и абсолютной ошибок вычислений, соответственно, `1e-5` и `1e-8`. Используйте опцию численной процедуры `Jumps` для задания точки нарушения непрерывности при $t = -10^{-6}$. Обеспечьте

прерывание выполнения программы с использованием функции обработки события в случае, если величина $y_4(t) - 0.1$ стала равна нулю. Используйте значение $+1$ для параметра `state` в ситуации, когда $y_4(t) \leqslant 0.1$ и -1 при $y_4(t) > 0.1$. Задача должна быть решена при $y_4(0) = 0$, поэтому начальное значение параметра `state` равно $+1$. При каждом выходе из численной процедуры интегрирования выполняйте проверку на достижение конца интервала интегрирования. Если `sol.x(end)<60`, изменяйте знак параметра `state` и запускайте заново процедуру `dde23`, используя полученное решение в качестве функции истории. В функции вычисления системы ДУЗА положите $\xi(y_4(t)) = 1$, если `state` равно $+1$ и $\xi(y_4(t)) = \frac{10}{9}[1 - y_4(t)]$ — в противном случае. Помните, если вы передаете параметр `state` как опциональный аргумент численной процедуры `dde23`, вы должны указать этот параметр в списке входных аргументов процедуры вычисления функции истории.

■ УПРАЖНЕНИЕ 4.9

В работе [Hale, 1971] рассматриваются модели систем «хищники–жертвы», в случае, когда существует ограничение на доступные ресурсы для роста популяции жертв и рождаемость хищников изменяется в зависимости от изменения численности популяций жертв y_1 и хищников y_2 лишь после некоторого временно́го запаздывания τ. В условиях этих предположений с использованием простейшей модели [Ortega & Poole, 1981] в виде системы ОДУ

$$y_1'(t) = ay_1(t) + by_1(t)y_2(t),$$
$$y_2'(t) = cy_2(t) + dy_1(t)y_2(t),$$

получаем более сложную систему с запаздываниями

$$y_1'(t) = ay_1(t)\left(1 - \frac{y_1(t)}{m}\right) + by_1(t)y_2(t),$$
$$y_2'(t) = cy_2(t) + dy_1(t-\tau)y_2(t-\tau).$$

Интересно исследовать влияние величины запаздывания на динамику системы. Для этого решим обе системы на интервале $[0, 100]$ с начальными значениями $y_1(0) = 80$ и $y_2(0) = 30$; в случае решения системы ДУЗА эти же значения выступают в роли постоянной функции истории. Предположим, что значения параметров следующие

$$a = 0.25,\ b = -0.01,\ c = -1.00,\ d = 0.01,\ m = 200.$$

Напомним, что систему ОДУ можно решать с использованием численной процедуры `dde23`, задавая пустой вектор запаздываний `[]`. В число входных аргументов процедуры для вычисления дифференциальных уравнений должна входить переменная `z`. Если массив запаздываний является пустым, численная процедура `dde23` вызывает эту функцию с входными аргументами, в число которых также входит `z`, что может быть использовано при ее программной реализации. Если `isempty(Z)=TRUE`, значит решается система ОДУ, соответствующая первой задаче, в противном случае — решается вторая задача. Таким образом, в одной процедуре можно реализовать вычисление двух систем дифференциальных уравнений. При

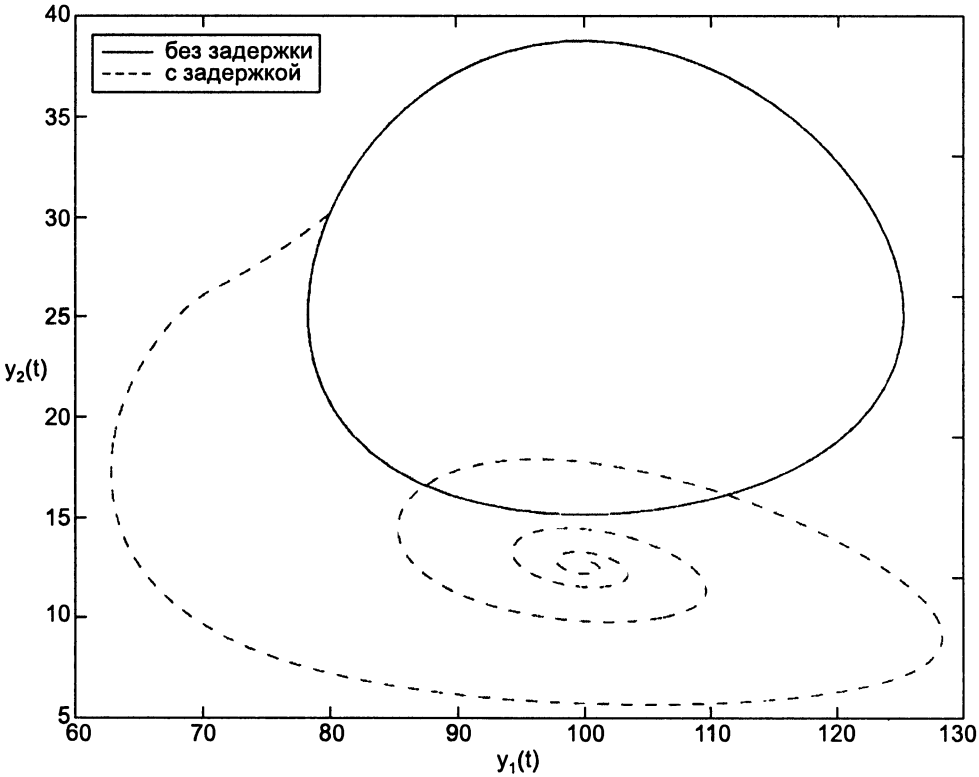

Рис. 4.7. Модель «хищники–жертвы» при наличии временно́го запаздывания и без такового.

стандартном подходе можно использовать две различные функции, предназначенные соответственно для вычисления первой системы ОДУ и второй системы ДУЗА. Решите систему ДУЗА при $\tau = 1$. Выведите на одном графике кривые $(y_1(t), y_2(t))$ для первой и второй систем. Фазовый портрет решения системы ОДУ должен представлять собой замкнутую кривую, соответствующую предельному циклу системы. Для получения этого результата необходимо ужесточить требования на допустимые ошибки вычислений. Например, при

```
options = ddeset('RelTol',1e-5,'AbsTol',1e-8);
```

можно получить фазовый портрет, изображенный на рисунке 4.7. Для получения фазового портрета в виде замкнутой кривой возможно потребуется выполнить несколько численных экспериментов при последовательно уменьшающихся значениях допустимой величины ошибки.

Из рисунка 4.7 видно, что введение в модель временно́го запаздывания может привести к существенному изменению качественных характеристик поведения решения. Выполнив численные эксперименты при различных значениях τ, вы сможете убедиться в том, что это утверждение остается верным даже при введении малых запаздываний. Интересно также удалить из системы член, отвечающий за

ограничение ресурсов для развития популяции жертв $1 - y_1(t)/m$ и исследовать изменение орбиты при изменении τ.

■ УПРАЖНЕНИЕ 4.10

В работе [Ottesen, 1997] рассматривается модель сердечно-сосудистой системы со следующими переменными: артериальное давление $P_a(t) = y_1(t)$, венозное давление $P_v(t) = y_2(t)$ и частота сердцебиений $H(t) = y_3(t)$. В указанной работе исследовались условия, при которых наличие запаздывания приводит к качественному изменению решения и, в частности, к колебаниям компоненты $P_a(t)$. Система исследовалась при запаздываниях $\tau = 1.0,\ 1.4,\ 3.9,\ 5.0,\ 7.5$ и 10. Вычислите решения и выведите график артериального давления при значениях запаздывания τ равной 1.0 и 7.5, когда решения претерпевают особенно заметные изменения. Решите на интервале $[0, 350]$ систему уравнений

$$\begin{aligned}
y_1'(t) &= -\frac{1}{c_a R} y_1(t) + \frac{1}{c_a R} y_2(t) + \frac{1}{c_a} V_{str} y_3(t), \\
y_2'(t) &= \frac{1}{c_v R} y_1(t) - \left(\frac{1}{c_v R} + \frac{1}{c_v r}\right) y_2(t), \\
y_3'(t) &= f(T_s, T_p),
\end{aligned}$$

где

$$\begin{aligned}
T_s &= \left[1 + \left(\frac{y_1(t-\tau)}{\alpha_s}\right)^{\beta_s}\right]^{-1}, \\
T_p &= \left[1 + \left(\frac{\alpha_p}{y_1(t)}\right)^{\beta_p}\right]^{-1}, \\
f(T_s, T_p) &= \frac{\alpha_H T_S}{1 + \gamma_H T_p} - \beta_H T_p,
\end{aligned}$$

При $t \leqslant 0$ решение постоянно

$$\begin{aligned}
y_1(t) &= P_0, \\
y_2(t) &= \left(\frac{r}{r+R}\right) P_0, \\
y_3(t) &= \left(\frac{1}{r+R}\right)\left(\frac{P_0}{V_{str}}\right).
\end{aligned}$$

В указанной работе использовались значения констант

$$c_a = 1.55,\ c_v = 519,\ R = 1.05,\ r = 0.068,\ \alpha_0 = \alpha_s = \alpha_p = 93,\ \alpha_H = 0.84$$

и

$$\beta_0 = \beta_S = \beta_p = 7,\ \beta_H = 1.17,\ \gamma_H = 0,\ V_{str} = 67.9,\ P_0 = 93.$$

На одном из рисунков, представленных в [Ottesen, 1997], показаны графики решения, соответствующие ситуации, когда, начиная с момента $t = 600$, периферическое давление (peripheral pressure) R экспоненциально падает с постоянного

значения $R = 1.05$ до постоянного значения $R = 0.84$. Изменение R приводит к отчетливо прослеживаемому изменению частоты сердцебиений. В рассматриваемом случае величина запаздывания была равна $\tau = 4$ и интегрирование выполнялось на интервале $[0, 1000]$. Модифицируйте вашу программу так, чтобы можно было вывести график изменения частоты сердцебиений. Для этого (1) задайте значение опции `Jumps` равным 600, что приведет к соответствующей настройке алгоритма численной процедуры на случай, когда имеет место нарушение непрерывности производных решения малого порядка в известный момент времени, и (2) модифицируйте функцию вычисления ДУЗА, добавив следующие операторы

```
if t <= 600
   R = 1.05;
else
   R = 0.21 * exp(600-t) + 0.84;
end
```

■ УПРАЖНЕНИЕ 4.11

Модель Планта для взаимодействия нейронов (см. [MacDonald, 1989]) имеет вид системы уравнений

$$y_1'(t) = y_1(t) - \frac{y_1^3(t)}{3} - y_2(t) + m(y_1(t-\tau) - y_{1,0}),$$
$$y_2'(t) = r(y_1(t) + a - by_2(t)),$$

где $y_{1,0}$ — первая компонента стационарного решения $(y_{1,0}, y_{2,0})$. При значениях параметров

$$a = 0.8,\ b = 0.7,\ r = 0.08$$

решите эту систему уравнений на интервале $[0, 60]$ с функцией истории

$$y_1(t) = 0.4y_{1,0},\ y_2(t) = 1.8y_{2,0}.$$

Для проведения численных экспериментов в указанной работе было выбрано единственное вещественное стационарное решение $y_{1,0}$, удовлетворяющее $y_{1,0}^2 > 1 - rb$. Тогда после выполнения некоторых аналитических исследований можно воспользоваться MATLAB-функцией `roots` и вычислить значение $y_{1,0} = -1.22764016121492$. Используйте это значение $y_{1,0}$ для нахождения $y_{2,0}$. Найдите решение системы ДУЗА при $\tau = 20$ для двух случаев $m = +10$ и $m = -10$. Выведите график решений и убедитесь в том, что они существенно отличаются друг от друга.

■ УПРАЖНЕНИЕ 4.12

Модель популяции, описанная в работе [Cooke, van den Driessche & Zou, 1999], позволяет исследовать эффект воздействия запаздывания полового созревания и нелинейности закона изменения уровня рождаемости на увеличение общей численности особей. Модель имеет вид следующего ДУЗА

$$y'(t) = be^{-ay(t-T)}y(t-T)e^{-d_1T} - dy(t),$$

где $y(t)$ — численность популяции. Это уравнение должно быть решено на интервале $[0, 25]$ с функцией истории $y(t) = 3.5$ при $t \leqslant 0$. Подобные задачи обычно решаются при нескольких наборах значений параметров, и читателю предлагается сделать это для следующих четырех случаев:

1. $a = 1,\ d = 1,\ d_1 = 1,\ b = 20$;
2. $a = 1,\ d = 1,\ d_1 = 1,\ b = 80$;
3. $a = 1,\ d = 1,\ d_1 = 0,\ b = 20$;
4. $a = 1,\ d = 1,\ d_1 = 0,\ b = 80$.

Это упражнение может быть выполнено при четырех отдельных запусках соответствующей программы для каждого из наборов параметров. Но можно написать такую программу, которая позволяет выполнить все вычисления при одном запуске. Для этого следует определить массив `B=[20 80 20 80]` и решить рассматриваемое ДУЗА в цикле `for` по индексу `dataset` с параметром b, равным `b=(dataset)`. Для каждого из наборов параметров решите ДУЗА при трех значениях запаздывания $T = 0.2$, 1.0 и 2.4 и выведите графики полученных решений на одном рисунке. Вы заметите, что величина запаздывания существенным образом влияет на поведение решения. Заметим, что структура-решение может иметь вид индексированного массива, что позволяет в рассматриваемом примере упростить программный код. Определим массив `Delays = [0.2 1.0 2.4]`. Тогда вычисление решений для этих запаздываний может быть выполнено с использованием следующих операторов

```
for i = 1:3
    T = Delays(i);
    sol(i) = dde23(@ddes,T,3.5,[0, 25],opts);
end
```

После выхода из цикла узлы сетки и решение уравнения для первого значения запаздывания могут быть получены в виде полей решения-структуры `sol(1).x` и `sol(1).y` и аналогично для других значений запаздывания. Значение T должно быть передано в процедуру `ddes` как параметр или как глобальная переменная, т. к. эта величина входит в ДУЗА. В представленном фрагменте программного кода указанная величина и набор параметров передаются в процедуру как глобальные переменные. При численном решении поставленной задачи требования на допустимую величину ошибки вычислений должны быть более жесткими, по сравнению с принятыми по умолчанию: соответствующие величины `RelTol=1e-5` и `AbsTol=1e-8` должны быть заданы с использованием функции `ddeset`.

■ УПРАЖНЕНИЕ 4.13

В работе [Martin & Ruan, 2001] рассмотрен вопрос о влиянии запаздывания и постоянного роста численности жертв на решение нескольких известных моделей «хищники–жертвы». В этом упражнении вы должны численно исследовать представленные в указанной статье модели. Во всех случаях величины $x(t)$ и $y(t)$ представляют собой соответственно число хищников и жертв в момент времени t.

- Первая модель, рассмотренная в вышеуказанной статье, характеризуется тем, что запаздывание имеет место в члене, соответствующем увеличению

численности жертв. В этом случае рассматриваемая система имеет следующий вид

$$x'(t) = x(t)\left[2\left(1 - \frac{x(t-\tau)}{40}\right) - \frac{y(t)}{x(t)+10}\right] - 10,$$
$$y'(t) = y(t)\left[\frac{x(t)}{x(t)+10} - \frac{2}{3}\right].$$

Покажите, что точка $(20, 15)$ является точкой равновесия системы. При $\tau = 0$ решите эту систему ОДУ с начальными данными $x(0) = 40$ и $y(0) = 16$ на интервале $[0, 100]$. (Для этого используйте численную процедуру `dde23` с пустым массивом `lags`.) С использованием графика численного решения $(x(t), y(t))$ убедитесь, что точка равновесия является асимптотически устойчивой при нулевом значении запаздывания τ. При $\tau = 0.826$ решите соответствующую систему ДУЗА на $[0, 100]$ с постоянной функцией истории $x = 40$ и $y = 2$. Выведите график $(x(t), y(t))$ и на основании этих численных результатов убедитесь, что при указанном значении запаздывания τ в системе имеет место предельный цикл. В обоих случаях при выполнении численного интегрирования уменьшите допустимое значение относительной ошибки вычислений `RelTol` до величины 10^{-5}.

- Вторая модель, рассмотренная в вышеуказанной статье, характеризуется тем, что запаздывание имеет место в функции отклика, определяющей изменение числа хищников в ответ на изменение численности жертв. В этом случае рассматриваемая система имеет следующий вид

$$x'(t) = x(t)\left[2\left(1 - \frac{x(t)}{50}\right) - \frac{y(t)}{x(t)+40}\right] - 10,$$
$$y'(t) = y(t)\left[-3 + \frac{6x(t-\tau)}{x(t-\tau)+40}\right].$$

Покажите, что точка $(40, 12)$ является точкой равновесия системы. Для каждого из значений запаздывания $\tau = 7$ и $\tau = 9$ решите соответствующую систему ДУЗА на $[0, 250]$ с постоянной функцией истории $x = 44$ и $y = 2$. Выведите графики $(x(t), y(t))$ и на основании этих численных результатов убедитесь, что при значении запаздывания $\tau = 7$ точка равновесия является асимптотически устойчивой, а при значении запаздывания $\tau = 9$ в системе имеет место предельный цикл. В обоих случаях при выполнении численного интегрирования уменьшите допустимое значение относительной ошибки вычислений `RelTol` до величины 10^{-5}.

- Наконец, третья модель, рассмотренная в вышеуказанной статье, использовалась для исследования изменения решений известной модели «хищники–жертвы» после добавлении в нее члена, соответствующего постоянному росту численности жертв. В этом случае рассматриваемая система имеет следующий вид

$$x'(t) = x(t)[20 - x(t) - y(t)] - 7,$$
$$y'(t) = -15y(t) + 3x(t-\tau)y(t-\tau).$$

Покажите, что точка $(5, \frac{68}{5})$ является точкой равновесия системы. При $\tau = 0.05$ решите систему ДУЗА на интервале $[0, 10]$ с постоянной функцией истории $x = 2$ и $y = 10$. Выведите график $(x(t), y(t))$ и на основании этих численных результатов убедитесь, что в системе имеет место предельный цикл. При выполнении численного интегрирования уменьшите допустимое значение относительной ошибки вычислений `RelTol` до величины 10^{-5}.

■ УПРАЖНЕНИЕ 4.14

В работе [Marriott & DeLisle, 1989] рассмотрена задача о контролере. Соответствующее ДУЗА включает ступенчатую функцию от решения, вычисленную с запаздыванием: при $\Delta = y(t-12) - x_b$ исследуемое уравнение имеет вид

$$y'(t) = \tau^{-1}\left(-y(t) + \pi(a + \varepsilon\,\mathrm{sign}(\Delta) - u\sin 2(\Delta))\right).$$

Это уравнение решается на интервале $[0, 120]$ с функцией истории $y(t) = 0.6$ при $t \leqslant 0$ и при значениях параметров

$$x_b = -0.427,\ a = 0.16,\ \varepsilon = 0.02,\ u = 0.5,\ \tau = 1.$$

Численная процедура `dde23` достаточно робастна для рассматриваемого случая и может быть использована для решения этой задачи без какого-либо доопределения функции $\mathrm{sign}(\Delta)$. Тем не менее, соответствующие нарушения непрерывности могут создать определенные проблемы при численном решении ОДУ [Shampine & Thompson, 2000] и в меньшей степени при численном решении ДУЗА. Для того, чтобы быть более уверенным в достоверности полученных численных результатов, следует предпринять определенные шаги для того, чтобы обеспечить гладкость правой части соответствующего дифференциального уравнения. Это может быть сделано по аналогии с тем, как это было сделано в программе `ch4ex5a.m`: путем задания значения параметра `state`, используемого численной процедурой в качестве значения $\mathrm{sign}(\Delta)$. Для рассматриваемой в задаче функции истории этот параметр инициализируется значением $+1$. Используйте опцию `Events` для прерывания процесса интегрирования в случае, если $y(t-12) - x_b = 0$. Если это терминальное событие происходит до момента достижения конечной точки интервала интегрирования, измените знак `state` и заново запустите процедуру численного интегрирования на интервале `[sol.x(end),120]`, используя полученную структуру-решение в качестве параметра, определяющего функцию истории на новом интервале интегрирования. Обеспечьте вывод на экран компьютера соответствующих сообщений о возникновении подобных событий, а также начальных точек для каждого нового этапа численного интегрирования.

§ 4.5. ДРУГИЕ ТИПЫ ДУЗА И ЧИСЛЕННЫЕ МЕТОДЫ ИХ РЕШЕНИЯ

Ранее в этой главе рассматривались уравнения

$$y'(t) = f(t, y(t), y(t-\tau_1), y(t-\tau_2), \ldots, y(t-\tau_k)) \tag{4.7}$$

с постоянными запаздываниями τ_j. В более общем случае запаздывания в этом уравнении могут зависеть от независимой переменной t. При условии, что эти запаздывания отделены от нуля, подобные уравнения могут быть решены аналогично

тому, как это делается в случае, когда запаздывания являются постоянными. Однако анализ распространения нарушений непрерывности становится более трудной задачей. Нарушение непрерывности производной малого порядка в t^* «проявляется» в уравнении (4.7), когда $t - \tau_j(t) = t^*$. Иными словами, для каждого запаздывания нарушение непрерывности распространяется до момента, соответствующего выполнению равенства

$$t - \tau_j(t) - t^* = 0. \tag{4.8}$$

Если запаздывания постоянны, решение этого уравнения представляет собой тривиальную задачу и соответствует случаю распространения нарушения непрерывности до момента $t^* + \tau_j$. Если же запаздывания не являются постоянными, решение этого уравнения должно находиться численно. В некоторых случаях эта задача может быть достаточно трудной. Например, непросто вычислить нулевое решение четной кратности. Кроме того, задача нахождения местоположения любого нулевого решения кратности больше единицы плохо определена. Если запаздывания могут обращаться в ноль, возникают дополнительные проблемы, носящие концептуальный и вычислительный характер. Например, в ДУЗА

$$y'(t) = y\left(\frac{t}{2}\right) = y\left(t - \frac{t}{2}\right),$$

определенном при $t \geqslant 0$, не используются значения переменной $y(t)$, вычисляемые при $t < t_0 = 0$; иными словами, в этом ДУЗА не используется функция истории. В случаях, когда величина запаздывания меньше шага интегрирования, в численной процедуре `dde23` используются неявные формулы и итеративные методы, что позволяет повысить ее вычислительную эффективность. Эта особенность программной реализации алгоритма процедуры `dde23` приобретает особенное значение при обращающихся в ноль запаздываниях. Робастные численные процедуры для нахождения нулей функций остаются работоспособными и в случаях, когда эти функции не являются гладкими, но следует иметь ввиду, что эти процедуры наиболее эффективны при нахождении нулей гладких функций. В целях иллюстрации проявления двух вышеуказанных проблем, рассмотрим уравнение

$$y'(t) = y(t)(1 - y([t])),$$

где $[t]$ — функция наибольшего целого числа, т. е. $[t] = n$ при $n \leqslant t < n + 1$. Таким образом запаздывание $\tau(t) = t - [t]$ характеризуется нарушением непрерывности и обращается в ноль при целых значениях t.

Если запаздывания зависят от решения уравнения, ситуация еще более усложняется. Так же как и в случае постоянных запаздываний, если запаздывания зависят лишь от времени, уравнение распространения (4.8) может быть решено перед выполнением процедуры численного интегрирования. Однако в рассматриваемой ситуации уравнение распространения принимает вид

$$t - \tau_j(t, y(t)) - t^* = 0$$

и для ответа на вопрос о распространении запаздываний необходимо знать $y(t)$. Очевидно, что решение этого уравнения необходимо выполнять совместно

с решением рассматриваемого дифференциального уравнения и с использованием неявных итеративных методов. На практике решение этого уравнения необходимо получать на каждом шаге. В некоторых численных процедурах в качестве необходимой оценки решения $y(t)$ на предполагаемом шаге интегрирования используется экстраполяция решения, вычисленного на текущем шаге. С использованием этого предсказанного решения выполняется проверка на наличие нарушений непрерывности на протяжении предполагаемого шага интегрирования, при необходимости этот шаг интегрирования уменьшается и, наконец, применяется итеративный метод. Эта процедура вычислительно очень затратна. Кроме того, на сегодняшний день некоторые аспекты ее программной реализации не могут быть признаны удовлетворительными. В качестве примера рассмотрим уравнение

$$y_1'(t) = y_2(t),$$
$$y_2'(t) = y_2(e^{1-y_2(t)})y_2^2(t)e^{1-y_2(t)}.$$

Это уравнение решается на интервале $[0.1, 5]$ с функцией истории, полученной с использованием аналитического решения: $y_1(t) = \ln(t)$ и $y_2(t) = t^{-1}$. Заметим, что запаздывание не только зависит от решения $t - e^{(1-y_2(f))}$, но и обращается в ноль при $t = 1$.

Выше были рассмотрены уравнения, в которых производная решения в текущий момент времени t зависела от решения, вычисленного в этот момент времени, а также в одном или нескольких предыдущих моментах времени. Однако в более общем случае эта производная может также зависеть и от *производной* решения, вычисленной в эти предыдущие моменты времени. Подобные уравнения называются *уравнениями нейтрального типа*. В качестве примера рассмотрим

$$y'(t) = \cos(t)[1 + y(ty^2(t))] + 0.3y(t)y'(ty^2(t)) + \\ + 0.7\sin(t)\cos(t\sin 2(t)) - \sin(t + t\sin 2(t)). \qquad (4.9)$$

Это уравнение решается на интервале $[0, \frac{\pi}{2}]$ с функцией истории, определяемой из аналитически полученного решения $y(t) = \sin(t)$. Решение подобных ДУЗА является непростой задачей и разработка соответствующей теории далека от завершения. Одной из основных трудностей является то обстоятельство, что для подобных систем не существует развитой теории о существовании, единственности и непрерывной зависимости решения от начальных данных и параметров. В отличие от ранее рассмотренных случаев ДУЗА, нарушения непрерывности производных не «сглаживаются» в процессе интегрирования. Все эти обстоятельства приводят к возникновению определенных трудностей при разработке эффективных процедур численного интегрирования уравнений нейтрального типа. Очевидно, что в процессе интегрирования необходимо сохранять в оперативной памяти компьютера вычисленные приближенные значения производных решения. К сожалению, при использовании непрерывных обобщений численных методов, позволяющих получить непрерывную аппроксимацию $S(t)$ решения $y(t)$ определенного порядка точности, соответствующая аппроксимация $S'(t)$ производной $y'(t)$ обычно имеет меньший порядок точности.

Несмотря на наличие указанных выше трудностей, существующие численные процедуры решения ДУЗА, обсуждаемые ниже, позволяют успешно решить

Таблица 4.1: Статистика численной процедуры DKLAG6 при решении ДУЗА нейтрального типа при обращающимся в ноль запаздывании

Tol	Steps	Evals	EROI	ERO2	Ratio
10^{-2}	2	50	0.191e-02	0.191e-02	1.00
10^{-4}	3	77	0.298e-01	0.464e-01	1.55
10^{-6}	6	212	0.283e+00	0.308e+00	1.09
10^{-8}	11	347	0.524e-01	0.187e+00	3.57
10^{-10}	21	554	0.242e+00	0.157e+01	6.48

уравнения нейтрального типа. В качестве иллюстрации решим уравнение (4.9) с использованием DKLAG6. Кроме наличия фундаментальной проблемы, заключающейся в том, что производная решения в текущий момент времени зависит от производной решения в предыдущие моменты времени, рассматриваемое уравнение характеризуется еще и тем, что запаздывание зависит от t и y и обращается в ноль в начальный и конечный моменты времени, что представляет собой дополнительную трудность. В таблице 4.1 представлены некоторые статистические данные, касающиеся процесса интегрирования этого уравнения. Аналогичные результаты можно получить при использовании любой из рассмотренных ниже численных процедур. В таблице приведены значения следующих переменных:

- Tol — допустимая величина ошибки;
- Steps — число шагов интегрирования;
- Evals — число вычислений производных;
- EROI — отношение величины максимальной ошибки к допустимой величине ошибки в каждом из узлов сетки;
- ERO2 — отношение величины максимальной ошибки к допустимой величине ошибки в каждой точке интервала интегрирования;
- Ratio — отношение максимальной величины глобальной ошибки интерполяции к максимальной величине глобальной ошибки интегрирования.

Несмотря на то, что описанные в этом параграфе задачи находятся в настоящее время лишь в стадии интенсивного исследования, уже существуют достаточно эффективные численные процедуры для их решения. В первую очередь следует отметить упомянутые выше процедуры ARCHI [Paul, 1995] и DKLAG6 [Corwin et al., 1997; Corwin & Thompson, 1996], а также DDVERK [Enright & Hayashi, 1997, 1998]. Все три процедуры предназначены для решения уравнений нейтрального типа и уравнений с запаздываниями, зависящими от решения, являющимися малыми или обращающимися в ноль. Эти процедуры основаны на использовании явных формул Рунге–Кутта, допускают непрерывное обобщение и реализованы на языке Fortran 77.

Численная процедура ARCHI основана на использовании (4,5)-пары формул Рунге–Кутта. Отметим, что отслеживание нарушения непрерывности является опциональной особенностью этой программы. Контроль локальной ошибки в явных

методах Рунге–Кутта достаточно робастен, что позволяет выявить и локализовать нарушения непрерывности и на основании полученной информации заново выполнить вычисления при новом значении шага интегрирования. Использование контроля величины локальной ошибки представляется привлекательным подходом, поскольку в этом случае для выполнения вычислений требуется минимальное вмешательство пользователя. С другой стороны, по сравнению с непосредственным отслеживанием нарушений непрерывности этот подход менее надежен и может привести к потере точности вычислений. Численная процедура ARCHI предоставляет пользователю возможность выбора метода вычисления неявных формул — либо путем экстраполяции, либо с использованием итеративной процедуры.

Численная процедура DDVERK основана на использовании (5,6)-пары формул. Ситуации с малыми и обращающимися в ноль запаздываниями обрабатываются итеративно. Нарушения непрерывности выявляются при контроле ошибки вычислений. Точки возможного возникновения нарушений непрерывности локализуются и при их прохождении используются специальные интерполяционные полиномы.

Численная процедура DKLAG6 также основана на использовании (5,6)-пары формул и отслеживание нарушения непрерывности является опциональной особенностью этой программы. DKLAG6 — это единственная численная процедура, допускающая использование аппарата локализации событий. Интерфейс пользователя существенным образом отличается от интерфейса других численных процедур — взаимодействие с этой численной процедурой осуществляется через специальные функции, программно реализованные пользователем. Для автоматического генерирования текстов этих функций, а также текстов функций вывода полученных решений, существует специальное программное средство (см. [Corwin & Thompson, 1993]).

ЛИТЕРАТУРА

1. W. Ames & E. Lohner (1981). Nonlinear models of reaction-diffusion in rivers. In R.Vichnevetsky & R. Stepleman (Eds)., *Advances in Computer Methods for Partial Differential Equations,* vol. IV, pp. 217-19. New Brunswick, NJ: IMACS.
2. P. Amodio, J. R. Cash, G. Roussos, R.W. Wright, G. Fairweather, I. Gladwell, G. L. Kraut & M. Paprzycki (2000). Almost block diagonal linear systems: Sequential and parallel solution techniques, and applications. *Numer. Lin. Alg. Appl.* 7: 275-317.
3. D. Arnold & J. C. Polking (1999). *Ordinary Differential Equations Using MATLAB,* 2nd ed. Englewood Cliffs, NJ: Prentice-Hall.
4. U.M. Ascher, J. Christiansen & R. D. Russell (1979). COLSYS - A collocation code for boundary value problems. In B. Childs et al. (Eds.), *Codes for Boundary Value Problems* (Lecture Notes in Comput. Sci., 76), pp. 164-85. NewYork: Springer-Verlag.
5. U. M. Ascher, J. Christiansen & R. D. Russell (1981). Collocation software for boundary value ODE's. *ACM Trans. Math. Software* 7: 209-29.
6. U. M. Ascher, R. M. M. Mattheij & R. D. Russell (1995). *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations.* Philadelphia: SIAM.
7. U. M. Ascher & R. D. Russell (1981). Reformulation of boundary value problems into «standard» form. *SIAM Review* 23: 238-54.
8. G. Bader & U. Ascher (1987). A new basis implementation for a mixed order boundary value solver. *SIAM J. Sci. Stat. Comput.* 9: 483-500.
9. P. B. Bailey, B. S. Garbow, H. G. Kaper & A. Zettl (1991). Eigenvalue and eigenfunction computations for Sturm-Liouville problems. *ACM Trans. Math. Software* 17: 491-9.
10. P. B. Bailey, M. K. Gordon & L. F. Shampine (1978). Automatic solution of the Sturm-Liouville problem. *ACM Trans. Math. Software* 4: 193-208.
11. P. B. Bailey, L. F. Shampine & P. E.Waltman (1968). *Nonlinear Two Point Boundary Value Problems.* NewYork: Academic Press.
12. C. T. H. Baker, C.A. H. Paul & D. R.Wille (1995a). A bibliography on the numerical solution of delay differential equations. Numerical Analysis Report no. 269, Mathematics Department, University of Manchester, U.K.
13. C. T. H. Baker, C. A. H. Paul & D. R.Wille (1995b). Issues in the numerical solution of evolutionary delay differential equations. *Adv. Comput. Math.* 3: 171-96.
14. C. M. Bender & S. A. Orszag (1999). *Advanced Mathematical Methods for*

Scientists and Engineers I, Asymptotic Methods and Perturbation Theory. NewYork: Springer-Verlag.
15. P. Bogacki & L. F. Shampine (1989). A 3(2) pair of Runge-Kutta formulas. *Appl. Math. Lett.* 2: 1-9.
16. R. L. Borrelli & C. S. Coleman (1999). *ODE Architect.* NewYork:Wiley.
17. R.W. Brankin, J. R. Dormand, I. Gladwell, P. Prince & W. L. Seward (1989). ALGORITHM 670: A Runge-Kutta-Nystrom code. *ACM Trans. Math. Software* 15: 31-40.
18. R.W. Brankin & I. Gladwell (1994). A Fortran 90 version of RKSUITE: An ODE initial value solver. *Ann. Numer. Math.* 1: 363-75.
19. R.W. Brankin, I. Gladwell & L. F. Shampine (1993). RKSUITE:A suite of explicit Runge-Kutta codes. In R. P. Agarwal (Ed.), *Contributions to Numerical Mathematics* (WSSIAA, 2), pp. 41-53. Singapore: World Scientific.
20. K. E. Brenan, S. L. Campbell & L. R. Petzold (1996). *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations* (SIAM Classics in Applied Mathematics, 14). Philadelphia: SIAM.
21. P. N. Brown, G. D. Byrne & A. C. Hindmarsh (1989).VODE: A variable coefficient ODE solver. *SIAM J. Sci. Stat. Comput.* 10: 1038-51.
22. J. R. Cash & M. H.Wright (1991). A deferred correction method for nonlinear two-point boundary value problems: Implementation and numerical evaluation. *SIAM J. Sci. Stat. Comput.* 12: 971-89.
23. T. K. Caughy (1970). Large amplitude whirling of an elastic string - A nonlinear eigenvalue problem. *SIAM J. Appl. Math.* 18: 210-37.
24. T. Cebeci & H. B. Keller (1971). Shooting and parallel shooting methods for solving the Falkner-Skan boundary-layer equations. *J. Comp. Phys.* 7: 289-300.
25. J. D. Cole (1968). *Perturbation Methods in Applied Mathematics.*Waltham, MA: Blaisdell.
26. K. Cooke, P. van den Driessche & X. Zou (1999). Interaction of maturation delay and nonlinear birth in population and epidemic models. *J. Math. Biol.* 39: 332-52.
27. S. P. Corwin, D. Sarafyan & S. Thompson (1997). DKLAG6: A code based on continuously imbedded sixth order Runge-Kutta methods for the solution of state dependent functional differential equations. *Appl. Numer. Math.* 24: 319-33.
28. S. P. Corwin & S. Thompson (1993). DRAKE: Continuous simulation software for the solution of delay differential equations on personal computers. Computer Science Department Technical Report Series, no. TR-93-001, Radford University, Radford,VA.
29. S. P. Corwin & S. Thompson (1996). DKLAG6: Solution of systems of functional differential equations with state dependent delays. Computer Science Department Technical Report Series, no. TR-96- 002, Radford University, Radford, VA.
30. A. R. Curtis, M. J. D. Powell & J. K. Reid (1974). On the estimation of sparse Jacobian matrices. *J. Inst. Math. Appl.* 13: 117-19.
31. H. T. Davis (1962). *Introduction to Nonlinear Differential and Integral Equations.* NewYork: Dover.
32. F. R. de Hoog & R.Weiss (1976). Difference methods for boundary value problems with a singularity of the first kind. *SIAM J. Numer. Anal.* 13: 775-813.
33. F. R. de Hoog & R.Weiss (1978). Collocation methods for singular boundary

value problems. *SIAM J. Numer. Anal.* 15: 198-217.
34. J. R. Dormand (1996). *Numerical Methods for Differential Equations*. Boca Raton, FL: CRC Press.
35. J. R. Dormand & P. J. Prince (1980). A family of embedded Runge-Kutta formulae. *J. Comput. Appl. Math.* 27: 19-26.
36. C. H. Edwards (1997). Newton's nose-cone problem. *Mathematica J.* 7: 64-71.
37. W. H. Enright & H. Hayashi (1997). A delay differential equation solver based on a continuous Runge- Kutta method with defect control. *Numer. Algorithms* 16: 349-64.
38. W. H. Enright & H. Hayashi (1998). Convergence analysis of the solution of retarded and neutral differential equations by continuous methods. *SIAM J. Numer. Anal.* 35: 572-85.
39. W. H. Enright & P. H.Muir (1996). Runge-Kutta software with defect control for boundary value ODEs. *SIAM J. Sci. Comput.* 17: 479-97.
40. J. D. Farmer (1982). Chaotic attractors of an infinite-dimensional dynamical system. *Physica D* 4: 366- 93.
41. E. Fehlberg (1970). Klassiche Runge-Kutta-Formeln vierter und niedrigerer Ordnung mit Schrittenweiten- Kontrolle und ihre Anwendung aufWarmeleitungsprobleme. *Computing* 6: 61-71.
42. B. A. Finlayson (1972). *The Method of Weighted Residuals and Variational Principles*. NewYork: Academic Press.
43. C. A. J. Fletcher (1983). *Computational Galerkin Methods*. NewYork: Springer-Verlag. *GAMS. The Guide to Available Mathematical Software* is available at `http://gams.nist.gov/`
44. C. W. Gear (1971). *Numerical Initial Value Problems in Ordinary Differential Equations*. Englewood Cliffs, NJ: Prentice-Hall.
45. L. Genik & P. van den Driessche (1999). An epidemic model with recruitment-death demographics and discrete delays. In S. Ruan, G. S. K.Wolkowicz & J.Wu (Eds.), *Differential Equations with Applications to Biology,* pp. 237-49. Providence, RI: American Mathematical Society.
46. F. R. Giordano & M. D. Weir (1991). *Differential Equations: A Modeling Approach*. Reading, MA: Addison-Wesley.
47. I. Gladwell (1979a). The development of the boundary-value codes in the ordinary differential equations chapter of the NAG library. In B. Childs et al. (Eds.), *Codes for Boundary Value Problems* (Lecture Notes in Computer Science, 76), pp. 122-43. NewYork: Springer-Verlag.
48. I. Gladwell (1979b). Initial value routines in the NAG library. *ACM Trans. Math. Software* 5: 386-400.
49. I. Gladwell (1987). The NAG library boundary value codes. Numerical Analysis Report no. 134, Department of Mathematics, University of Manchester, U.K.
50. H2KL. The *Harwell 2000 Library*, at `hsl.rl.ac.uk`.
51. E. Hairer, S. P. Norsett & G.Wanner (1987). *Solving Ordinary Differential Equations I.* Berlin: Springer-Verlag.
52. E. Hairer & G.Wanner (1991). *Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems*. Berlin: Springer-Verlag.
53. J. Hale (1971). *Functional Differential Equations*. Berlin: Springer-Verlag.

54. P. Henrici (1962). *Discrete Variable Methods in Ordinary Differential Equations.* NewYork:Wiley.
55. P. Henrici (1977). *Error Propagation for Difference Methods.* NewYork: Krieger.
56. D. J. Higham & N. J. Higham, *MATLAB Guide.* Philadelphia: SIAM.
57. A. C. Hindmarsh & G. D. Byrne (1976). Applications of EPISODE: An experimental package for the integration of systems of ordinary differential equations. In L. Lapidus & W. E. Schiesser (Eds.), *Numerical Methods for Differential Systems*, pp. 147-66. NewYork: Academic Press.
58. M. H. Holmes. *Introduction to Perturbation Methods.* NewYork: Springer-Verlag.
59. T. E. Hull,W. H. Enright, B. M. Fellen & A. E. Sedgwick (1972). Comparing numerical methods for ordinary differential equations. *SIAM J. Numer. Anal.* 9: 603-37.
60. T. E. Hull,W. H. Enright & K. R. Jackson (1975). User's guide for DVERK — A subroutine for solving non-stiff ODEs. Report no. 100, Computer Science Department, University of Toronto, Ontario.
61. IMSL (2002). *The IMSL FORTRAN 77 Mathematics and Statistics Libraries (FNL),* ver. 3.0.Visual Numerics Inc., Houston, TX.
62. E. Isaacson & H. B. Keller (1966). *Analysis of Numerical Methods.* NewYork:Wiley.
63. E. Kamke (1971). *Differentialgleichungen Losungsmethoden und Losungen,* vol. I. NewYork: Chelsea.
64. H. B. Keller (1992). *Numerical Methods for Two-Point Boundary-Value Problems.* NewYork: Dover.
65. J. Kierzenka (1998). Studies in the numerical solution of ordinary differential equations. Doctoral dissertation, Department of Mathematics, Southern Methodist University, Dallas, TX.
66. J. Kierzenka & L. F. Shampine (2001). A BVP solver based on residual control and the Matlab PSE. *ACM Trans. Math. Software* 27: 299-316.
67. H. Kocak (1989). *Differential and Difference Equations through Computer Experiments.* New York: Springer-Verlag.
68. M. Kubiček, V. Hlavaček & M. Holodnick (1979). Test examples for comparison of codes for nonlinear boundary value problems in ordinary differential equations. In B. Childs et al. (Eds.), *Codes for Boundary-Value Problems in Ordinary Differential Equations* (Lecture Notes in Computer Science, 76), pp. 325-46. NewYork: Springer-Verlag.
69. J. D. Lambert (1991). *Numerical Methods for Ordinary Differential Systems.* NewYork:Wiley.
70. L. Lapidus, R. C. Aiken & Y. A. Liu (1973). The occurrence and numerical solution of physical and chemical systems having widely varying time constants. In R. A.Willoughby (Ed.), *Stiff Differential Systems,* pp. 187-200. NewYork: Plenum.
71. H. T. Laquer & B.Wendroff (1981). Bounds for the model quench front. *SIAM J. Numer. Anal.*18: 225-41.
72. M. Lentini & V. Pereyra (1974).A variable order finite difference method for nonlinear multipoint boundary value problems. *Math. Comp.* 23: 981-1003.

J. Lighthill (1986). *An Informal Introduction to Theoretical Fluid Mechanics.* Oxford: Clarendon.
73. C. C. Lin & L.A. Segel (1998). *Mathematics Applied to Deterministic Problems in the Natural Sciences.* Philadelphia: SIAM.
74. N. MacDonald (1978). *Time Lags in Biological Models.* Berlin: Springer-Verlag.
75. N. MacDonald (1989). *Biological Delay Systems: Linear Stability Theory.* Cambridge University Press.
76. Maple (1998). *Maple V Release 6.*Waterloo Maple Inc.,Waterloo, Ontario.
77. M. Marletta & J. D. Pryce (1995). LCNO Sturm-Liouville problems — Computational difficulties and examples. *Numer. Math.* 69: 303-20.
78. C. Marriott & C. DeLisle (1989). Effects of discontinuities in the behavior of a delay differential equation. *Physica D* 36: 198-206.
79. A. Martin & S. Ruan (2001). Predator-prey models with delay and prey harvesting. *J. Math. Biol.* 43: 247-67.
80. Matlab (2000). *MATLAB 6.* The MathWorks, Inc., Natick, MA.
81. R. M. M. Mattheij & G.W. M. Staarink (1984a). An efficient algorithm for solving general linear twopoint BVP. *SIAM J. Sci. Stat. Comput.* 5: 745-63.
82. R.M.M. Mattheij & G.W.M. Staarink (1984b). On optimal shooting intervals. *Math. Comp.* 42: 25-40.
83. C. B. Moler (1997). Are we there yet? *MATLAB Newsletter* (Simulink 2 Special Edition), pp. 16-17; see `http://www.mathworks.com/company/newsletter/pdf/97slCleve.pdf`.
84. C. B. Moler & L. P. Solomon (1970). Integrating square roots. *Comm. ACM* 13: 556-7.
85. J. S. Murphy (1965). Extensions of the Falkner-Skan similar solutions to flows with surface curvature. *AIAA J.* 3: 2043-9.
86. J. D. Murray (1993). *Mathematical Biology,* 2nd ed. Berlin: Springer-Verlag.
87. NAG (2002). *NAG FORTRAN 77 Library,* mark 21. Numerical Algorithms Group Inc., Oxford, U.K.
88. *Netlib.* The *Netlib* software repository is available at `http://www.netlib.org/`.
89. K.W. Neves (1975). Automatic integration of functional differential equations: An approach. *ACMTrans. Math. Software* 1: 357-68.
90. K.W. Neves & S. Thompson (1992). Software for the numerical solution of systems of functional differential equations with state dependent delays. *Appl. Numer. Math.* 9: 385-401.
91. H. J. Oberle & H. J. Pesch (1981). Numerical treatment of delay differential equations by Hermite interpolation. *Numer. Math.* 37: 235-55.
92. R. E. O'Malley (1991). *Singular Perturbation Methods for Ordinary Differential Equations.* NewYork: Springer-Verlag.
93. J. M. Ortega & W. G. Poole (1981). *An Introduction to Numerical Methods for Differential Equations.* Marshfield, MA: Pitman.
94. J. T. Ottesen (1997). Modelling of the baroflex-feedback mechanism with time-delay. *J. Math. Biol.* 36: 41-63.
95. C.A. H. Paul (1995). A user-guide to ARCHI. Numerical Analysis Report no. 283, Mathematics Department, University of Manchester, U.K.

96. S. Pruess, C. T. Fulton & Y. Xie (1992). Performance of the Sturm-Liouville software SLEDGE. Technical Report no. MCS-91-19, Department of Mathematical Sciences, Colorado School of Mines, Golden.
97. J. D. Pryce (1993). *Numerical Solution of Sturm-Liouville Problems.* Oxford: Clarendon.
98. J. D. Pryce (1999). A test package for Sturm-Liouville solvers. *ACM Trans. Math. Software* 25: 21-57.
99. A. Raghothama & S. Narayanan (2002). Periodic response and chaos in nonlinear systems with parametric excitation and time delay. *Nonlinear Dynam.* 27: 341-65.
100. S. M. Roberts & J. S. Shipman (1972). *Two-Point Boundary Value Problems: Shooting Methods.* New York: Elsevier.
101. H. H. Robertson (1996). The solution of a set of reaction rate equations. In J.Walsh (Ed.), *Numerical Analysis: An Introduction*, pp. 178-82. London: Academic Press.
102. J. M. Sanz-Serna & M. P. Calvo (1994). *Numerical Hamiltonian Problems.* London: Chapman & Hall.
103. M. R. Scott (1973). *Invariant Imbedding and Its Applications to Ordinary Differential Equations, An Introduction.* Reading, MA: Addison-Wesley.
104. R. Seydel (1988). *From Equilibrium to Chaos.* NewYork: Elsevier.
105. L. F. Shampine (1986). Conservation laws and the numerical solution of ODEs. *Comput Math. Appl.* 12B: 1287-96.
106. L. F. Shampine (1994). *Numerical Solution of Ordinary Differential Equations.* NewYork: Chapman & Hall.
107. L. F. Shampine (1998). Linear conservation laws for ODEs. *Comput Math. Appl.* 35: 45-53.
108. L. F. Shampine (2002).Variable order Adams codes. *Comput Math. Appl.* 44: 749-61.
109. L. F. Shampine, R. C. Allen, Jr. & S. Pruess (1997). *Fundamentals of Numerical Computing.* NewYork: Wiley.
110. L. F. Shampine, I. Gladwell & R.W. Brankin (1991). Reliable solution of special root finding problems for ODEs. *ACM Trans. Math. Software* 17: 11-25.
111. L. F. Shampine & M. K. Gordon (1975). *Computer Solution of Ordinary Differential Equations.* San Francisco: Freeman.
112. L. F. Shampine & M.W. Reichelt (1997). The Matlab ODE suite. *SIAM J. Sci. Comput.* 18: 1-22.
113. L. F. Shampine, M. W. Reichelt & J. A. Kierzenka (1999). Solving index-1 DAEs in Matlab and Simulink. *SIAM Review* 41: 538-52.
114. L. F. Shampine & S. Thompson (2000). Event location for ordinary differential equations. *Comput.Math. Appl.* 39: 43-54.
115. L. F. Shampine & S. Thompson (2001). Solving DDEs in Matlab. *Appl. Numer. Math.* 37: 441-58.
116. R. D. Skeel & M. Berzins (1990). A method for the spatial discretization of parabolic equations in one space variable. *SIAM J. Sci. Stat. Comput.* 11: 1-32.
117. S. S. Soliman & M. D. Srinath (1998). *Continuous and Discrete Signals and Systems,* 2nd ed. Englewood Cliffs, NJ: Prentice-Hall.

118. H. J. Stetter (1973). *Analysis of Discretization Methods for Ordinary Differential Equations.* NewYork: Springer-Verlag.
119. A. M. Stuart & A. R. Humphries (1996). *Dynamical Systems and Numerical Analysis.* Cambridge University Press.
120. S. Suherman, R. H. Plaut, L. T.Watson & S. Thompson (1997). Effect of human response time on rocking instability of a two-wheeled suitcase. *J. Sound Vibration* 207: 617-25.
121. L. Tavernini (1996). *Continuous-Time Modeling and Simulation.* Amsterdam: Gordon & Breach.
122. S. Thompson & P. G. Tuttle (1986). Benchmark fluid flow problems for continuous simulation languages. *Comput. Math. Appl.* 12A: 345-52.
123. L. N. Trefethen (2000). *Spectral Methods in MATLAB.* Philadelphia: SIAM.
124. D.A.Wells (1967). *Theory and Problems of Lagrangian Dynamics* (Schaum's Outline Series). NewYork: McGraw-Hill.
125. D. R.Wille & C. T. H. Baker (1992). DELSOL - A numerical code for the solution of systems of delaydifferential equations. *Appl. Numer. Math.* 9: 223-34.
126. S. J. Wolfram (1996). *The Mathematica Book*, 3rd ed. Wolfram Media & Cambridge University Press.
127. Марчук Г.И. Математические модели в иммунологии // Вычислительные методы и эксперименты. 3-е изд., перераб. и доп. — М.: Наука, 1991.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

Л. Ф. ШАМПАЙН, И. ГЛАДВЕЛ, С. ТОМПСОН

РЕШЕНИЕ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ С ИСПОЛЬЗОВАНИЕМ MATLAB

Учебное пособие

Корректор *А. М. Плетнева*
Подготовка иллюстраций *В. В. Воскресенская*
Выпускающие *Е. А. Антипова, О. В. Шилкова*

ЛР № 065466 от 21.10.97
Гигиенический сертификат 78.01.07.953.П.004173.04.07
от 26.04.2007 г., выдан ЦГСЭН в СПб

Издательство «ЛАНЬ»
lan@lpbl.spb.ru; www.lanbook.com
192029, Санкт-Петербург, Общественный пер., 5.
Тел./факс: (812)412-29-35, 412-05-97, 412-92-72.
Бесплатный звонок по России: 8-800-700-40-71

Подписано в печать 20.07.09.
Бумага офсетная. Гарнитура Литературная. Формат 70×100 $^{1}/_{16}$.
Печать офсетная. Усл. п. л. 24,70. Тираж 1500 экз.

Заказ № 3558.

Отпечатано в полном соответствии с качеством
предоставленных диапозитивов в ОАО «Дом печати — ВЯТКА».
610033, г. Киров, ул. Московская, 122
Факс: (8332) 25-58-83, 53-53-80
http://www.gipp.kirov.ru
e-mail: pto@gipp.kirov.ru

ГДЕ КУПИТЬ

ДЛЯ ОРГАНИЗАЦИЙ:

Для того, чтобы заказать необходимые Вам книги, достаточно обратиться в любую из торговых компаний Издательского Дома «ЛАНЬ»:

по России и зарубежью

«ЛАНЬ-ТРЕЙД»
192029, Санкт-Петербург, ул. Крупской, 13
тел.: (812) 412-85-78, 412-14-45, 412-85-82
тел./факс: (812) 412-54-93
e-mail: trade@lanpbl.spb.ru
ICQ: 446-869-967
www.lanpbl.spb.ru/price.htm

в Москве и в Московской области

«ЛАНЬ-ПРЕСС»
109263, Москва, 7-ая ул. Текстильщиков, д. 6/19
тел.: (499) 178-65-85
e-mail: lanpress@ultimanet.ru

в Краснодаре и в Краснодарском крае

«ЛАНЬ-ЮГ»
350072, Краснодар, ул. Жлобы, д. 1/1
тел.: (8612) 74-10-35
e-mail:lankrd98@mail.ru

ДЛЯ РОЗНИЧНЫХ ПОКУПАТЕЛЕЙ:

интернет-магазины:
«Сова»: http://www.symplex.ru
«Ozon.ru»: http://www.ozon.ru
«Библион»: http://www.biblion.ru

также Вы можете отправить заявку
на покупку книги по адресу:
192029, Санкт-Петербург, ул. Крупской, 13

Издательство «ЛАНЬ»

Мы будем благодарны Вам
за пожелания по издаваемой нами литературе,
а также за предложения по изданию книг
новых авторов или переизданию
уже существующих трудов.

Мы заинтересованы в сотрудничестве
с высшими учебными заведениями
и открыты для Ваших предложений
по улучшению нашего взаимодействия.

Теперь Вы можете звонить нам бесплатно
из любых городов России по телефону

8-800-700-40-71

Дополнительную информацию
и ответы на вопросы Вы также можете получить,
обратившись по электронной почте:

market@lpbl.spb.ru

МАРЧУК Г. И.

МЕТОДЫ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ

УЧЕБНОЕ ПОСОБИЕ

В учебном пособии рассмотрены методы построения разностных схем для дифференциальных уравнений; интерполяция сеточных функций; методы решения стационарных и нестационарных задач математической физики; методы Шварца и разделения области; методы возмущений; методы оптимизации; повышение точности приближенных решений. Основное внимание уделяется сложным задачам математической физики, которые в процессе решения сводятся, как правило, к более простым, допускающим реализацию алгоритмов на ЭВМ. Рассмотрены многие современные подходы к численным методам.

Учебное пособие предназначено для студентов старших курсов и аспирантов по специальности «Прикладная математика», также может быть полезно для научных работников в области вычислительной математики.