Х.Гулд, Я.Тобочник

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ В ФИЗИКЕ: ЧАСТЬ 1.

Книга авторов из США предназначена для обучения читателя моделированию физических экспериментов на компьютере (и тем самым обучению физике). В первой части основное внимание уделено детерминированным системам. Каждая глава содержит теоретический материал, методы решения соответствующих задач, тексты программ, задачи и контрольные вопросы. В основном изложении используется True Basic, в приложении программы приведены на Паскале и Фортране-77; здесь же дан справочный материал, облегчающий перенос программ на различные модели компьютеров. Может служить учебным пособием.

Для студентов физических и технических вузов, аспирантов, преподавателей физики, молодых специалистов.

ОГЛАВЛЕНИЕ

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

ПРЕДИСЛОВИЕ ПЕРЕВОДЧИКОВ

Уважаемый потенциальный Читатель, взяв в руки эту книгу, возможно, Вы скривите губы скептической улыбкой и подумаете: «Сколько можно писать книг о численном моделировании». Хотелось бы предостеречь Вас от возможного скепсиса. Дело в том, что, несмотря на избитое название, Вы держите в руках необычную книгу. В ней авторы попытались внедрить принципы компьютерного мышления в изучение физики. Иначе говоря, заставить компьютер исследовать разнообразные физические процессы. Методическая основа такого подхода понятна — чтобы учить других, надо знать самому. Необходимость получить от машины ответ на поставленный вопрос требует от читателя более глубокого проникновения в суть изучаемой проблемы, полученный ответ порождает новые вопросы, и в результате происходит закрепление теоретического материала и развивается физическая интуиция. В этой ситуации компьютер выступает в роли экспериментальной установки для проведения «физических» опытов, причем читатель должен сам провести эксперимент и проинтерпретировать полученные результаты.

Насколько нам известно, издание такого рода книги в нашей стране не является первым опытом в этой области. В отличие от большинства учебников, посвященных численному моделированию конкретных физических систем, в представленных двух томах авторы делятся опытом использования компьютера в описанном выше контексте для изучения широкого класса физических задач от падения тел в поле тяжести Земли, колебаний, простейших задач теплопроводности до таких современных разделов, как хаос в динамических системах, задачи перколяции, метод ренорм-группы и исследование полимеров методами случайного блуждания, которые почти не освещаются в учебной литературе на русском языке.

Поскольку читатель должен использовать в вычислительном эксперименте не чужие программы, а свои модифицированные варианты базовых программ из текста, чтение книги потребует от него заметных усилий. В качестве основного языка программирования в книге используется True Basic. Ввиду того что этот язык в нашей стране почти не используется, адаптация программ для вариантов «уличного» Бейсика потребует от советского читателя дополнительных усилий. Для тех, кто знаком с языками Паскаль или Фортран-77, в конце каждого тома авторы привели распечатки основных программ, встречающихся в тексте. Каждый читатель волен выбрать наиболее подходящий ему способ

изучения этой книги. Резюмируя все сказанное выше, можно сказать, что изучение этой книги потребует от Вас немало усилий, которые с лихвой окупятся тем удовлетворением, которое Вы испытаете после ее прочтения, сродни тому, которое получили переводчики в процессе работы.

Обратной стороной широты излагаемых в книге физических вопросов является некоторая поверхностность изложения теоретического материала. Но, несмотря на этот недостаток, мы надеемся, что книга в первую очередь будет полезна студентам физических и других специальностей в качестве дополнения к курсу общей физики и частично к ряду спецкурсов, читаемых на старших курсах, а преподаватели найдут в ней ряд интересных задач и методических приемов.

В заключение необходимо сказать несколько слов о литературе. Поскольку книга рассчитана на студентов вузов, а вся цитируемая литература — англоязычная (за редким исключением), мы посчитали возможным привести дополнительный список литературы с краткими аннотациями, в которой на русском языке рассматриваются вопросы, затронутые в книге. Мы надеемся, что все эти книги имеются в любой вузовской библиотеке и будут доступны заинтересованному читателю. Список дополнительной литературы построен не в алфавитном порядке, а по принципу: от простого — к сложному.

А. Полюдов
В. Панченко

Памяти Чен-кенг Ма

ПРЕДИСЛОВИЕ

Численное моделирование составляет неотъемлемую часть современной фундаментальной и прикладной науки, причем по важности оно приближается к традиционным экспериментальным и теоретическим методам. Поэтому умение «вычислять» входит в обязательный репертуар научных работников и преподавателей.

Философия данной книги хорошо выражается китайской пословицей (источник нам неизвестен):

«Я слышу и забываю, я вижу и запоминаю, я делаю и постигаю.»

Нас интересует не как можно использовать компьютер для обучения физике, а как можно научить студентов обучать компьютер. Наша основная цель — создать такие условия, в которых читатель научит компьютер моделировать физические системы. Как показывает наш опыт, активное участие в численном моделировании вырабатывает более глубокое интуитивное понимание физических концепций. Другие задачи настоящей книги — познакомить с методами молекулярной динамики и Монте-Карло, собрать простые, но реалистичные в научном плане задачи в один курс для начинающих, а также научить на примерах методам структурного программирования.

Значительная часть материала, вошедшего в этот учебник, была использована в односеместровом курсе под названием «Лаборатория численного моделирования», предлагавшемся в университете Кларка. Этому курсу предшествует односеместровая подготовка по физике и математическому анализу. Предварительные знания по программированию для ЭВМ необязательны. Курс прошли студенты младших и старших курсов, специализирующиеся по физике, химии, биологии, математике, информатике, географии и электротехнике. Уровень их подготовки по физике и программированию был самым разнообразным — от минимального до высокого. Выяснилось, что проведение многих численных экспериментов вполне доступно слушателям с ограниченной подготовкой по физике, а более сильным студентам удается углубить свои знания по вопросам, которые они уже изучали. Данный курс организован аналогично другим лабораторным курсам в университете Кларка, с двумя лекционными занятиями в неделю, на которых излагается основной материал и производится опрос студентов. Знакомство с методами программирования осуществляется в ходе регулярно проводимых лабораторных занятий. Курс рассчитан на самостоятельную работу и предоставляет студентам

полную свободу в выборе оптимального темпа работы и задач, отвечающих их собственным интересам и подготовке.

Представляется, что численное моделирование играет уже достаточно важную роль, чтобы подобные курсы преподавались в других институтах. Однако с равным успехом эта книга могла бы найти и иное применение. Например, ее можно было бы использовать в качестве дополнения к вводному курсу физики для сильных студентов и в курсах промежуточного уровня по классической механике, волнам, электричеству и магнетизму, статистической физике и термодинамике, квантовой механике, а также по физической химии. Данное руководство может также служить основой курса по численным методам. Несмотря на то что книга начинается с основных понятий физики и дифференциального исчисления, мы считаем, что наиболее успешным будет использование этой книги на промежуточном уровне.

В качестве языка программирования в основном изложении используется True BASIC. В приложениях к каждой части книги приведены варианты некоторых программ на Паскале и Фортране-77. Мы сочли наиболее подходящим язык True BASIC, поскольку его легко выучить и использовать, в нем есть «настоящие» подпрограммы, отличные графические средства и к тому же он идет как на IBM PC и совместимых с ним компьютерах, так и на Apple Macintosh. Читатели, знакомые с «уличным» Бейсиком, не должны испытывать почти никаких трудностей по адаптации приведенных в тексте программ при условии, что они будут четко различать глобальные и локальные переменные. С нашей точки зрения, в Бейсике, Фортране и Паскале гораздо больше сходства, чем различий.

Думается, что вы сможете изучить программирование тем же путем, как это делали мы — вместе с определенным предметом. Хотя настоящая книга по возможности не ориентируется на какую-либо конкретную марку компьютера, мы настоятельно рекомендуем читателю с небольшим программистским опытом писать свои программы на персональном компьютере. Персональные компьютеры легче использовать, чем большие вычислительные установки, и, кроме того, они предоставляют легко доступные графические средства. Для решения приводимых в книге задач нужно в качестве справочников иметь под рукой руководство по программированию и учебник по физике.

Каждая глава содержит краткое обсуждение необходимых физических понятий, за которым следуют тексты программ, задачи и контрольные вопросы. Теоретический материал, программы и задачи взаимосвязаны, и поэтому обсуждаемые вопросы легче усваиваются после того, как все

задачи будут решены. Программные листинги следует рассматривать как текст, предназначенный для чтения, а не как исходный код для компьютера. Наши программы спроектированы так, чтобы они были простыми и легкочитаемыми, а не элегантными или эффективными. Для выполнения большинства заданий читатель должен понять логику соответствующих программ и тем самым логику отвечающей им физической системы. Большинство задач требует по крайней мере небольшой модификации программ. Мы считаем, что построчный ввод программ с клавиатуры поможет вам легче изучить программирование. Сами же программы в большинстве случаев короткие, и мы рекомендуем вам эти программы переделывать. Задачи организованы таким образом, чтобы очередная задача каждой главы служила основой для дальнейших задач этой и следующих глав. Задачи, отмеченные звездочкой, либо более сложные, либо требуют для своего решения значительно больше времени, чем средняя задача; их решение не является необходимым условием для работы с последующими главами. Приводимый в конце каждой главы список рекомендуемой литературы составлялся из соображений ее педагогической полезности, а не полноты или исторической точности. Мы просим простить наших коллег, чьи работы мы ненамеренно опустили, и будем всемерно признательны за предложения относительно новых и дополнительных ссылок.

В ч. 1 основной упор делается на классическую физику, а в ч. 2 на статистическую физику. Эти области отражают наши собственные научные интересы, и, кроме того, здесь легче всего можно познакомиться с методами численного моделирования. Мы рассматриваем также волны, оптические явления, электричество и магнетизм и квантовую механику. Каждая часть содержит материал, достаточный для семестрового курса по численному моделированию. В ч. 1 книги основное внимание уделяется моделированию детерминированных систем. В гл. 1 рассматривается применение компьютеров в физике и дается общая характеристика некоторых языков программирования. Гл. 2 знакомит с методом Эйлера численного интегрирования дифференциальных уравнений первого порядка. Поскольку многие читатели хорошо знакомы с методом Эйлера, главное назначение этой главы состоит в том, чтобы привести синтаксис основных конструкций языка True BASIC. Хотя для читателей, не обладающих опытом программирования, это введение, по-видимому, охватывает слишком обширный материал, в остальном тексте используется чрезвычайно мало дополнительных синтаксических конструкций. В гл. 3—5 модифицированный метод Эйлера используется для моделирования падения предметов, движения планет и колебательного движения. Кроме того, гл. 5 включает раздел по электрическим цепям и приложе-

ние с описанием других численных методов решения уравнений движения Ньютона. Гл. 6 знакомит с методом молекулярной динамики. Если читатель не имеет доступа к большому компьютеру (или следующему поколению микрокомпьютеров), то из данного моделирования можно получить только качественную картину тепловых процессов. Тем не менее этот метод важен в физике и химической физике, причем идеи, лежащие в его основе, являются непосредственным развитием предыдущих глав. Гл. 7 знакомит с нелинейными динамическими системами и возможностью использовать компьютер для «открытия« новых знаний. Гл. 8 и 9 содержат в основном традиционный материал по колебаниям и волнам, а также по электростатике и магнитостатике. Значительная часть материала этих двух глав ориентирована на визуальные демонстрации.

Все главы ч. 2 связаны с методом случайной выборки, обычно называемым методом Монте-Карло, применительно к задачам статистической физики и квантовой механики. В гл. 10 метод Монте-Карло вводится в связи с изучением численного интегрирования. Хотя в этой главе прямо и не обсуждаются физические явления, она позволяет рассмотреть различные методы на хорошо известном материале. Гл. 11 посвящена случайному блужданию и его приложению к физическим явлениям. В гл. 12—13 рассматриваются современные направления исследований, которые приобретают в наши дни все большее значение во многих областях науки. В этих главах мы обсуждаем перколяцию, простые идеи фазовых переходов и ренормгруппы, фракталы, законы локального роста и клеточные автоматы. Многие прикладные задачи имеют несложную постановку, но дают сложное поведение, которое выглядит весьма интересно. В гл. 14 рассматриваются задача о приближении к равновесному состоянию и методы вычисления энтропии. В гл. 15 с помощью относительно нового метода моделируется микроканонический ансамбль и «открывается» каноническое распределение Больцмана. Гл. 16 знакомит с методом Монте-Карло для моделирования тепловых систем. В гл. 17 мы обсуждаем моделирование квантовых систем методом Монте-Карло и более традиционными численными методами. В гл. 18 кратко обсуждается, как с помощью одних и тех же методов можно решать многие совершенно не связанные друг с другом задачи.

Широкое распространение персональных компьютеров и требования со стороны промышленности на специалистов, обладающих высокой компьютерной грамотностью, заставляют в настоящее время перестраивать учебные планы по всем дисциплинам с целью включения во все основные учебные курсы материала, связанного с компьютерами. До сих пор на большинстве физических факультетов компьютеры используются в каче-

стве инструмента для обработки данных и с демонстрационными целями. На некоторых физических факультетах читаются в настоящее время курсы по вычислительной физике и процессам измерения и управления. Однако их влияние, если его оценивать по углублению знаний студентов, росту числа студентов-физиков или изменениям в учебниках, еще весьма незначительно. Мы отдаем себе отчет в том, что организация нетривиального использования компьютеров в обучении физике займет многие годы. Надеемся, что настоящая книга внесет вклад в решение этой задачу, и с благодарностью примем ваши замечания, предложения и советы.

Мы внимательно проверили свои программы на предмет ошибок и опечаток. Однако наш опыт говорит о том, что почти не бывает раз и навсегда безошибочных программ. Поэтому мы не даем никаких гарантий, что приведенные в книге программы полностью свободны от ошибок. Преподаватели, которые захотят воспользоваться этой книгой для какого-нибудь курса, могут получить бесплатно дискету с программами (в формате IBM PC или Macintosh) через издательство Addison-Wesley.

БЛАГОДАРНОСТИ

Многие наши коллеги и студенты ознакомились с предварительными вариантами глав рукописи этой книги и высказали множество критических замечаний и советов, а также оказали нам общую поддержку. Особо хотелось бы поблагодарить Гарольда Абельсона, Даниеля Бен-Аврахама, Джона Дэвиса, Хьюга Де Уитта, Лизу Дандон, Ферридун Фамили, Джима Гивена, Джима Гантона, Мерилин Джименез, Габора Кальмана, Тома Кейза, Роберта Килмойера, Билла Клейна, Питера Клебана, Роджера Коина, Кристофера Ланди, Франсуа Лейвраза, Джона Махта, Джин Мазенко, Билла Мичалсона, Роберта Пелковитса, Джозефа Приста, Стена Райнака, Сидни Реднера, Питера Рейнолдса, Кун Ру, Дэвида Сторка, Ориола Валлса, Джералда Вишняка, Джорджа Вайсса, Питера Висшера и Ю-синг Янга. Само собой разумеется, что все ошибки, упущения и неясные места лежат целиком на нашей совести.

Курс, на котором базируется настоящая книга, нельзя было бы разработать без персональных компьютеров, подаренных университету Кларка фирмой IBM и небольшой субсидии Национального научного фонда на разработку курса. Кроме того, одному из авторов хотелось бы поблагодарить Комитет Меллона университета Кларка за поддержку в разработке учебного плана и фирму Digital Equipment Corporation за поддержку родственного проекта, предусматривающего включение компьютеров в учебный план для младшекурсников. Нам хотелось бы также отметить гостеприимство физического факультета Бостонского университета, где писались части этой книги.

Особую благодарность мы приносим Стейси Бресслеру, Шерри Хаулетт и Грегу Смедсраду из фирмы Apple Computer Corporation за интерес, проявленный к нашему курсу. Трудно себе представить, что эту книгу удалось бы написать, не будь у нас компьютера Macintosh и принтера Apple LaserWriter, подаренных нам фирмой Apple Computer Corporation. Огромной благодарности заслуживают Брюс Спац, бывший научный редактор издательства Addison-Wesley, за его поддержку, интерес и терпение, Шарон Ван-Ганди, указавшую нам на ряд грамматических ошибок, и Мона Зефтель за подготовку рукописи в виде оригинал-макета. Текст набирался с помощью редакторов TEX и MacWriter; рисунки изготовлены авторами с помощью пакетов MacPaint, MacDraw и Cricket Graph.

Мы признательны нашим женам Петти Гулд и Андреа Молл Тобочник, а также детям Гулда Джошуа, Эмили и Эвану за их мужество и понимание, проявленные в ходе работы над этой книгой.

ВВЕДЕНИЕ

1

Рассматриваются значение компьютеров в физике и природа численного моделирования.

1.1. ЗНАЧЕНИЕ КОМПЬЮТЕРОВ В ФИЗИКЕ

Достаточно пролистать первое попавшееся научное издание или пройтись по любой физической лаборатории, чтобы всюду встретить компьютеры. Говоря об использовании компьютеров в физике, можно выделить четыре категории:

1. Численный анализ.
2. Символьные преобразования.
3. Моделирование.
4. Управление в реальном времени.

В *численном анализе* вычислениям предшествует выяснение упрощающих физических принципов. Например, мы знаем, что решение многих физических задач может быть сведено к решению системы линейных уравнений. Рассмотрим уравнения

$$2x + 3y = 18,$$
$$x - y = 4.$$

Используя метод подстановки и карандаш с бумагой, легко найти *аналитическое* решение $x = 6$, $y = 2$. Предположим, что нам надо решить систему четырех уравнений. Мы в состоянии и на этот раз найти аналитическое решение, быть может, с помощью более сложного метода. Если же число переменных становится существенно бóльшим, нам приходится прибегать к численным методам и компьютеру и находить численное решение. В данном случае компьютер служит инструментом *численного анализа*, при этом в программу для компьютера закладываются все существенные физические принципы, например сведение рассматриваемой задачи к обращению матрицы. Поскольку часто бывает необходимо вычислить многомерный интеграл, произвести операции с большими матрицами или решить сложное дифференциальное уравнение, то понятно, что это применение компьютера играет в физике важную роль.

Менее известным, но приобретающим все большее значение применением компьютера в теоретической физике являются *аналитические преобразования*. В качестве примера предположим, что мы хотим узнать решение квадратного уравнения $ax^2 + bx + c = 0$. Программа аналитических преобразований может выдать решение в виде формулы $x = (-b \pm \pm\sqrt{b^2 - 4ac})/2a$. Кроме того, такая программа может выдать решения и в обычной числовой форме для конкретных значений a, b и c. С по-

мощью типичной программы аналитических преобразований можно выполнять такие математические операции, как дифференцирование, интегрирование, решение уравнений и разложение в степенной ряд. Чему же тогда будут учить, когда такие программы появятся на каждом персональном компьютере? Устареют ли таблицы интегралов так же, как это произошло с логарифмической линейкой?

Моделирование характеризуется тем, что в программу закладываются все основные законы модели с минимальным анализом. В качестве примера предположим, что каждому ученику в классе из 100 человек выдается по 10 долларов. Учительница, которая также начинает с 10 долларами в кармане, выбирает случайным образом ученика и бросает монету. Если выпадает «решка», учительница дает ученику 0.5 доллара; в противном случае ученик дает учительнице 0.5 доллара. Ни учительнице, ни ученику не разрешается делать долги. После большого числа обменов спрашивается: «Какова вероятность того, что у ученика имеется n долларов?» и «Какова вероятность того, что у учительницы имеется m долларов?» Одинаковы ли эти две вероятности? Один из способов найти ответы на указанные вопросы состоит в том, чтобы провести эксперимент. Однако такой эксперимент было бы затруднительно поставить и утомительно выполнять. Хотя данную конкретную задачу можно решить точно аналитическими методами, однако не все задачи удается решить таким способом. Можно поступить иначе, а именно заложить правила игры в программу для компьютера, промоделировать большое число обменов и вычислить вероятности. После получения численных значений вероятностей мы, возможно, по-новому посмотрим на их природу и их связь с обменом денег. Компьютер можно также использовать для выяснения вопросов типа «Что будет, если...?» Например, как бы изменились вероятности, если бы обмен производился по 1 доллару, а не по 0.5?

Если заменить игроков другими объектами (например, под деньгами понимать энергию) и слегка изменить правила игры, указанный тип моделирования может найти применение в задачах магнетизма и физики частиц (см. гл. 15). Использование компьютеров для моделирования в течение последних 25 лет помогло нам открыть новые упрощающие физические принципы.

При всем разнообразии использования компьютеров главной целью расчета является обычно «понимание, а не числа». Вычисления оказали глубокое влияние на образ наших занятий физикой, на характер важных вопросов в физике и на выбор нами физических систем для изучения. Все три способа использования компьютеров требуют по крайней мере

некоторых упрощающих приближений, позволяющих решить задачу численно. Однако, поскольку моделирование требует минимального предварительного исследования и выдвигает на первый план исследовательский режим учебы, мы выделяем в данной книге этот подход.

Компьютеры являются также важным инструментом в экспериментальной физике. Часто они связаны со всеми фазами лабораторного эксперимента, от проектирования аппаратуры, управления этой аппаратурой в ходе эксперимента и до сбора и анализа данных. Это привлечение вычислительной техники не только позволило экспериментаторам лучше спать по ночам, но сделало возможными эксперименты, которые иначе были бы неосуществимы. Некоторые из упомянутых задач, например проектирование аппаратуры или же анализ данных, близки к задачам, встречающимся в теоретическом расчете. Однако задачи, связанные с управлением и интерактивным анализом данных, качественно отличаются и требуют программирования в реальном времени и стыковки вычислительного оборудования с разнообразными типами установок, а потому рассмотрение возможностей использования компьютеров для управления в реальном времени следует искать в других книгах.

1.2. ПРИРОДА ЧИСЛЕННОГО МОДЕЛИРОВАНИЯ

Почему численное моделирование становится важным в физике в настоящее время? Одна из причин заключается в том, что большинство применяемых нами аналитических средств, таких как дифференциальное исчисление, больше всего подходит для исследования *линейных* задач. Например, вы, по-видимому, уже умеете анализировать движение частицы, подвешенной на пружинке, решая уравнение ее движения (второй закон Ньютона) в предположении линейной возвращающей силы. Однако множество природных процессов являются *нелинейными*, так что малые изменения в одной переменной могут привести скорее к большим, чем к малым изменениям в другой переменной. Поскольку нелинейные задачи удается решить аналитическими методами только в отдельных случаях, компьютер дает нам новый инструмент для исследования нелинейных явлений. Другая причина важности численного моделирования связана с тем, что мы интересуемся системами со многими степенями свободы или многими переменными. Примером такой задачи является случай с обменом денег, описанный в разд. 1.1.

Развитие компьютерной технологии приводит в наше время к новому взгляду на физические системы. Постановка вопроса: «Как можно сфор-

ТАБЛИЦА 1.1. Аналогии между вычислительным и лабораторным экспериментами

Лабораторный эксперимент	Вычислительный эксперимент
Образец	Модель
Физический прибор	Программа для компьютера
Калибровка	Тестирование программы
Измерение	Расчет
Анализ данных	Анализ данных

мулировать задачу на компьютере?» уже привела к новым формулировкам физических законов и осознанию того, что сколь практично, столь и естественно выражать научные законы в виде правил для компьютера, а не на языке дифференциальных уравнений. Сейчас этот новый взгляд на физические процессы приводит некоторых физиков к тому, чтобы рассматривать компьютер как некую физическую систему и разрабатывать новейшие архитектуры компьютеров, которые могут более эффективно моделировать природные физические системы.

Иногда численное моделирование называют *вычислительным экспериментом*, поскольку оно имеет очень много общего с лабораторными экспериментами. Некоторые аналогии показаны в табл. 1.1. Отправным пунктом численного моделирования является разработка идеализированной модели рассматриваемой физической системы. Затем нам необходимо определить процедуру или *алгоритм* для реализации данной модели на компьютере. Компьютерная программа моделирует физическую систему и описывает вычислительный эксперимент. Такой вычислительный эксперимент служит мостом между лабораторными экспериментами и теоретическими расчетами. Например, мы можем получить по существу точные результаты, моделируя идеализированную модель, у которой нет никакого лабораторного аналога. Сравнение результатов моделирования с соответствующими теоретическими расчетами служит стимулом развития вычислительных методов. С другой стороны, можно проводить моделирование на реалистичной модели с тем, чтобы осуществить более прямое сравнение с лабораторными экспериментами.

Численное моделирование, как и лабораторные эксперименты, не заменяет размышление, а является инструментом, который можно использовать для постижения сложных явлений. Но цель всех наших исследований фундаментальных явлений состоит в поиске таких объяснений физических явлений, которые можно записать на обратной стороне конверта или которые можно представить на пальцах!

1.3. ВАЖНОСТЬ ГРАФИКИ

Коль скоро сегодня компьютеры изменяют традиционные способы проведения физических исследований, они не могут не сказаться также на том, как изучать физику. Например, по мере того как компьютер будет играть все большую роль в нашем понимании физических явлений, визуальное представление сложных численных результатов будет приобретать даже еще бóльшую важность. Человеческий глаз вместе со способностью мозга к обработке изображений представляет собой очень сложное устройство для анализа видеоинформации. Большинство из нас могут очень быстро провести наилучшую прямую линию через последовательность экспериментальных точек. И такая прямая линия больше значит для нас, чем некая прямая «наилучшего приближения», нарисованная каким-нибудь статистическим пакетом, в котором мы не разбираемся. Наш глаз способен выделить структуры и тренды, быть может, сразу не заметные из таблиц данных, и в состоянии обнаружить изменения во времени, которые могут привести к пониманию важных механизмов, лежащих в основе поведения системы.

Тем не менее использование графических средств может улучшить наше понимание характера аналитических решений. Например, как вы представляете себе функцию синус? Вряд ли вы ответите, что это ряд, т.е. $\sin x = x - x^3/3! + x^5/5! + \ldots$; скорее вы скажете, что это график периодической функции постоянной амплитуды (рис. 1.1).

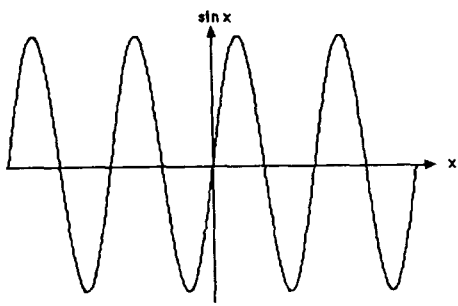

Рис. 1.1. График функции синус.

Здесь важно наглядное представление формы функции.

1.4. ЯЗЫКИ ПРОГРАММИРОВАНИЯ

В данной книге мы будем использовать три языка программирования: Бейсик, Фортран и Паскаль. Язык программирования Бейсик был разработан в 1965 г. Джоном Г. Кемени и Томасом Е. Курцом из Дармутского колледжа в качестве языка для вводных курсов по информатике. Паскаль был создан в 1970 г. Николасом Виртом из Швейцарского федерального технологического института и также был предназначен для целей обучения. Со временем оба языка были адаптированы для многих других целей. Фортран был разработан Джоном Бэкусом с коллегами в фирме IBM в период 1954–1957 гг. и является наиболее распространенным языком, применяемым в научных приложениях. Самая последняя версия известна как Фортран-77. Кроме того, применяются и многие другие языки программирования, например АПЛ, Си, Лисп, Модула-2, Пл/1.

Как нет наилучшего естественного языка, так нет и никакого одного наилучшего языка программирования. Языки программирования не являются чем-то застывшим, а продолжают изменяться с развитием аппаратных средств и теорий вычислений. Как это бывает с любым языком, практическое владение одним облегчает изучение другого. Бейсик обладает тем достоинством, что его легко выучить. К недостаткам Бейсика относятся его зависимость от аппаратуры и, что более важно, отсутствие средств модульного программирования. Паскаль хорошо структурирован и дает возможность описывать богатый набор структур данных. Кроме того, Паскаль налагает требования, которые затрудняют написание немодульных программ. Недостаток Паскаля состоит в том, что эти требования мешают быстро разрабатывать простые программы. Фортран допускает структурное программирование (речь идет только о Фортране-77), и его синтаксис во многом проще, чем у Паскаля. К основным недостаткам Фортрана относится то, что в настоящее время он не допускает рекурсивных подпрограмм и его типы данных более ограничены, чем в Паскале. Кроме того, такие управляющие структуры, как DO WHILE или DO UNTIL, не входят в стандарт Фортрана.

Новая версия Бейсика, названная True BASIC (Настоящий Бейсик), разработана недавно Кемени и Курцом. Язык True BASIC содержит подпрограммы подобно Фортрану и имеет отличные графические возможности, которые являются аппаратно независимыми. Одну и ту же программу, составленную на True BASIC, можно пропускать на столь разных компьютерах, как IBM PC и Apple Macintosh. Поскольку основной материал этой книги носит вводный характер и в связи с важностью графи-

ки, мы исходим из того, что большинство читателей будут разрабатывать свои программы на микрокомпьютерах. По этим причинам в качестве языка программирования в основном тексте книги мы приняли True BASIC. В конце каждой части книги приводятся версии большинства программ на Фортране и Паскале. Читатели, уже знающие какой-нибудь язык программирования, могут рассматривать листинги программ на True BASIC в качестве «псевдокода», который может быть легко переведен на тот язык, который им больше нравится. Читателям, изучающим язык программирования впервые, следует сравнить версии программ на разных языках, чтобы добиться пассивного понимания Фортрана и Паскаля. В конце ч. 1 в приложении А дается сравнительный синтаксис основных конструкций языков Microsoft BASIC, True BASIC, Фортран-77 и Паскаль. В приложении Б приведены примеры коротких программ.

1.5. ИЗУЧЕНИЕ ПРОГРАММ

Если вы уже умеете программировать, попробуйте прочитать какую-нибудь программу, написанную вами несколько лет (или даже несколько недель) назад. Для многих ученых оказалось бы не под силу разобраться в логике своих программ и им, следовательно, пришлось бы свои программы переписывать. Если вы только приступаете к изучению программирования, важно сразу выработать хорошие программистские привычки и избежать подобной трудности. В программах, представленных в этой книге, используются приемы *структурного* программирования, такие, например, как конструкции IF–THEN–ELSE и запрет на применение инструкций GOTO. Кроме того, программы написаны в виде *модулей*, которые представляют собой подпрограммы, выполняющие конкретные задачи.

В силу своего образования студенты естественнонаучных специальностей обладают особыми преимуществами при изучении программирования. Большинству из нас уже знакомы многочисленные кабели и другое оборудование, так что мы знаем, что ошибки в наших программах не смогут вывести из строя компьютер. Важнее то, что мы имеем уже готовый материал в конкретной предметной области, на котором можно осваивать программирование. Занятие физическими исследованиями с помощью компьютеров в последние несколько десятилетий дало нам многочисленные примеры, которые можно использовать для изучения физики, программирования и анализа данных. Поэтому мы призываем вас

изучать программирование на основе примеров, приведенных в каждой главе.

Наш опыт говорит о том, что единственным самым важным критерием качества прграммы является ее «читабельность». Если программу легко читать и понимать, то скорее всего это хорошая программа. Существует четкая аналогия между хорошей программой и хорошо написанным документом. Почти никогда программы не получаются идеальными с первого раза независимо от методов и правил, применяемых для их написания. Переписывание всегда будет важной частью программирования.

1.6. КАК ПОЛЬЗОВАТЬСЯ ЭТОЙ КНИГОЙ

Как правило, каждая глава начинается с короткого общего изложения характера рассматриваемой системы и важных общих вопросов. Затем мы знакомим читателя с вычислительными алгоритмами, при необходимости с синтаксисом True BASIC, и обсуждаем образец программы. Считается, что программы будут читаться как текст наравне с обсуждениями и задачами, разбросанными по всему тексту. Настоятельно рекомендуется прочитать все задачи, поскольку многие понятия вводятся после моделирования того или иного физического процесса.

Неплохо завести лабораторный журнал, куда можно записывать свои программы, результаты, выданные компьютером графики и свой анализ данных. Эта практика поможет вам выработать хорошие навыки для будущих исследовательских проектов, избежать дублирования, привести в порядок свои мысли и сэкономить время. В идеале журнал пригодится вам для написания лабораторного отчета или мини-исследования по своим программам, результатам, анализу данных и интерпретации.

ЛИТЕРАТУРА

Руководства по программированию

Мы советуем вам изучать программирование так же, как вы учили английский язык, — путем практики и прибегая чуть-чуть к помощи своих друзей и к руководствам. Мы приводим здесь список некоторых своих любимых руководств по программированию, но этот список ни в коей мере не является полным. Многие из руководств, которые снабжены языком программирования, также превосходны.

Richard E. Crandall, Pascal Applications for the Sciences, A Self-Teaching Guide, Wiley Press, 1984. Эта книга — не руководство по программированию, она содержит много примеров программ на Паскале и обсуждение природы Паскаля. Особенно интересны обсуждения программ трехмерной графики и быстрого преобразования Фурье — вопросов, которые не рассматриваются в данной книге.

Susan Finger, *Ellen Finger*, Advanced Applications for Introduction to Pascal with Applications in Science and Engineering, D.C. Heath, 1986.

John G. Kemeny, Thomas E. Kurtz, True BASIC, Addison-Wesley, 1985. Хорошее справочное руководство по True BASIC. Из других полезных книг по True BASIC назовем *William S. Davis*, True BASIC Primer, Addison-Wesley, 1986, и *Larry Joel Goldstein, C. Edward Moore, Peter J. Welcher*, Structured Programming with True BASIC, Prentice-Hall, 1986.

Vardell Lines, Pascal as a Second Language, Prentice-Hall, 1984. Если вы знаете структурное программирование, вы легко можете изучить другой язык.

Michael Metcalf, Effective FORTRAN 77, Clarendon Press, 1985. Отличная книга, правда, для подготовленных читателей. Автор приобрел богатый опыт в обработке больших объемов данных по физике высоких энергий.

Robert Moll, Rachael Folsom, Macintosh Pascal, Houghton Mifflin, 1985. Одна из многих дешевых книг по Паскалю.

Russ Walters, The Secret Guide to Computers, 12th ed., 1986. Трехтомный сборник одного из самых оригинальных писателей в промышленности компьютеров. Ранее вам еще не приходилось никогда читать подобное руководство по программированию. Попробуйте насладиться компьютерами и программированием так же, как Уолтерс. Наиболее интересен т. 3, в котором рассматриваются Фортран и Паскаль. Эти книги можно заказать по почте, написав Уолтеру по адресу 22 Ashland St., Somerville, MA 02144.

Robert Weiss, *Charles Seiter*, Pascal for FORTRAN Programmers, Addison-Wesley, 1984.

Общая литература по физике и компьютерам

Per Bak, Doing physics with microcomputers, Physics Today **36**, No. 12, pp. 25–29 (December, 1983).

Alfred Bork, Learning with Computers, Digital Press, 1981. Рас-

сматриваются разнообразные методы по применению компьютеров в обучении.

Robert Ehrlich, Physics and Computers, Houghton Mifflin, 1973. Эта книга и еще книга Гроссберга представляют собой отличные примеры пионерских учебников, которые внедряют применение компьютеров в курсы по физике для начинающих.

Robert G. Fuller, Resource letter CPE-1: Computers in physics education, Am. J. Phys. **54**, 782 (1986). Автор надеется, что, возможно, компьютер может внести больше радости в изучение физики.

Alan B. Grossberg, Fortran for Engineering Physics, McGraw-Hill, 1971. В этом лабораторном руководстве рассматриваются механические и тепловые системы.

Dieter W. Heermann, Computer Simulation Methods in Theoretical Physics, Springer-Verlag, 1986. Обсуждение методов молекулярной динамики и Монте-Карло, предназначенное для успевающих студентов младших курсов и начинающих студентов средних курсов.

Steven E. Koonin, Computational Physics, Benjamin/Cummings, 1986. Основной упор в этой книге сделан на применение численных методов. Приводится много нетривиальных задач.

John R. Merrill, Using Computers in Physics, University of Press of America, 1980. Вопросы, которые рассматривает автор и которые не отражены в нашей книге, включают приложения к релятивизму, ядерному распаду и физике твердого тела.

Herbert D. Peckham, Computers, BASIC, and Physics, Addison-Wesley, 1971. Почему пионерские книги начала 70-х годов не оказали большего влияния? Изменит ли что-нибудь повсеместная доступность микрокомпьютеров?

Commun. ACM **28**, No. 4, pp. 352-394 (April, 1985). Специальный раздел по вычислениям в теоретической физике.

Physics Today **36**, No. 5, pp. 24-62 (May, 1983). Специальный выпуск: "Doing Physics with Computers". Статьи *Donald R. Hamann*, "Computers in Physics: an overview"; *Michael Creutz*, "High-energy physics"; *Jorge E. Hirsch, Douglas J. Scalapino*, "Condensed-matter physics" и *Bruce I. Cohen, John Killeen*, "Computations in plasma physics".

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

Ниже приводится список литературы, рассчитанной на разный уровень подготовки читателей. Этот список не претендует на полноту. Читателю рекомендуется также, приступая к программированию для конкретного компьютера, ознакомиться с документацией по выбранному компилятору, уделив особое внимание отличиям языка реализации от стандарта.

Боон К., Паскаль для всех. — М.: Энергоатомиздат, 1988. Предназначена для первоначального знакомства с языком.

Геворкян Г. Х., *Семенов В. И.*, Бейсик это просто. — М.: Радио и Связь, 1989. Простая книга по «уличному» Бейсику.

Йенсен К., *Вирт Н.*, Практический курс языка Паскаль для микроЭВМ. — М.: Радио и Связь, 1986. Книга создателя языка Паскаль. Написана в стиле, присущем только Вирту.

Праттс Д., Программирование на языке Паскаль: практическое руководство. — М.: Мир, 1987. Книга рассчитана на читателя, знакомого с программированием.

Шаньгин В. Ф., Поддубная Л. М., Голубев—Новожилов Ю. М., Программирование на языке Паскаль. — М.: Высшая Школа, 1988.

Дьяконов В. П., Применение персональных ЭВМ и программирование на языке Бейсик. — М.: Радио и Связь, 1989.

Фокс А., *Фокс Д.*, Бейсик для всех: курс программирования на языке Бейсик для начинающих. — М.: Энергоатомиздат, 1986.

Уолш Б., Программирование на Бейсике. — М.: Радио и Связь, 1987.

Катцан Г., Язык программирования Фортран-77. — М.: Мир, 1982.

Колдербэнк В., Программирование на Фортране: Фортран-66 и Фортран-77. — М.: Радио и Связь, 1986.

ЗАДАЧА ОБ ОСТЫВАНИИ КОФЕ

2

В этой главе рассматривается простой метод численного решения дифференциальных уравнений и приводятся основные понятия программирования и графических методов.

2.1. ОСНОВНЫЕ ПОНЯТИЯ

Знакомство с численными методами неплохо начать с того, что расположиться подальше от компьютера и насладиться чашечкой горячего кофе (или чая). Однако, отхлебнув из чашечки кофе, мы по обыкновению обжигаемся, поскольку он очень горячий. Если нам не терпится, то можно добавить в кофе молока. Но если после этого кофе все еще горячий, ничего не остается делать, как подождать некоторое время, пока он не остынет до нужной температуры. Если желательно, чтобы кофе остыл как можно быстрее, то что лучше — добавить молоко сразу после приготовления кофе или немного подождать, прежде чем добавлять молоко?

Природа переноса тепла от кофе к окружающему пространству сложна и в общем случае включает в себя механизмы конвекции, излучения, испарения и теплопроводности. В том случае, когда разность температур между объектом и окружающей средой не очень велика, скорость изменения температуры объекта можно считать пропорциональной этой разности температур. Это утверждение более строго можно сформулировать на языке дифференциального уравнения:

$$\frac{dT}{dt} = -r(T - T_s), \tag{2.1}$$

где T — температура тела, T_s — температура окружающей среды, а r — «коэффициент остывания». Этот «коэффициент остывания» зависит от механизма теплопередачи, площади тела, находящегося в контакте со средой и тепловых свойств самого тела. Знак минус появляется в (2.1) во избежание нефизического эффекта увеличения температуры тела, когда $T > T_s$. Соотношение (2.1) называется *законом теплопроводности Ньютона*. Попытайтесь проинтегрировать уравнение (2.1) и получить зависимость температуры от времени.

Уравнение (2.1) — пример дифференциального уравнения *первого порядка*, поскольку в него входит только первая производная неизвестной функции $T(t)$. Ввиду того что множество процессов, происходящих в природе, описываются дифференциальными уравнениями, важно уметь решать эти уравнения. Рассмотрим уравнение первого порядка вида

$$\frac{dy}{dt} = g(x). \tag{2.2}$$

В общем случае *аналитического* решения уравнения (2.2), выраженного через хорошо известные функции, не существует. Кроме того, даже в том случае, когда аналитическое решение все же существует, необходимо представить решение в графическом виде, чтобы понять его характер. Эти причины побуждают нас искать не точные, а приближенные численные решения дифференциальных уравнений и познакомиться с простыми методами графического представления решений.

2.2. АЛГОРИТМ ЭЙЛЕРА

Типичный метод численного решения дифференциальных уравнений включает в себя преобразование дифференциального уравнения в *конечно-разностное*. Проанализируем уравнение (2.2). Положим, что при $x = x_0$ функция y принимает значение y_0. Поскольку уравнение (2.2) описывает изменение функции y в точке x_0, то можно найти *приближенное* значение функции y в близлежащей точке $x_1 = x_0 + \Delta x$, если приращение аргумента Δx мало. В первом приближении предполагается, что функция $g(x)$, или скорость изменения y, постоянна на отрезке от x_0 до x_1. В этом случае приближенное значение функции y в точке $x_1 = x_0 + \Delta x$ определяется выражением

$$y_1 = y(x_0) + \Delta y \approx y(x_0) + g(x_0)\Delta x. \qquad (2.3)$$

Мы можем повторить эту процедуру еще раз и найти значение y в точке $x_2 = x_1 + \Delta x$:

$$y_2 = y(x_1 + \Delta x) \approx y(x_1) + g(x_1)\Delta x. \qquad (2.4)$$

Очевидным образом это правило можно обобщить и вычислить приближенное значение функции в любой точке $x_n = x_0 + n\Delta x$ по итерационной формуле

$$y_n = y_{n-1} + g(x_{n-1})\Delta x \qquad (n = 0, 1, 2, \ldots). \qquad (2.5)$$

Данный метод называется методом касательных, или методом *Эйлера*. Можно предположить, что метод будет давать хорошее приближение к «истинному» значению функции y, если приращение аргумента Δx достаточно мало. Степень «малости» Δx определяется нашими требованиями и может не конкретизироваться до тех пор, пока метод не применяется для решения конкретных задач.

В методе Эйлера предполагается, что скорость изменения функции y на отрезке от x_{n-1} до x_n постоянна, а наклон касательной вычисляется в *начальной* точке отрезка. Графическая интерпретация выражения (2.5) приведена на рис. 2.1. Понятно, что в случае, когда наклон касательной меняется на некотором отрезке, появляется отклонение от точного решения. Тем не менее это отклонение можно уменьшить, если выбрать меньшее значение Δx.

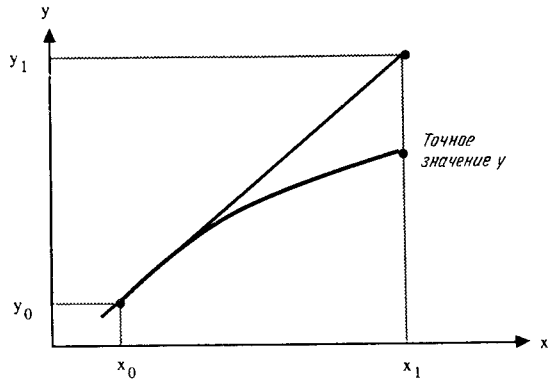

Рис. 2.1. Графическая интерпретация метода Эйлера. Наклон касательной вычисляется в начальной точке интервала. Приближению Эйлера и истинной функции соответствуют прямая и кривая.

2.3. ПРОСТОЙ ПРИМЕР

Чтобы решить задачу об остывании кофе в разд. 2.5, применим метод Эйлера для нахождения численного решения дифференциального уравнения $dy/dx = 2x$ с начальным условием $y = 1$ в точке $x = 1$. Мы хотим найти приближенное значение функции y в точке $x = 2$. Выбираем $\Delta x = 0.1$, тогда число шагов равно $n = (2 - 1)/\Delta x = 10$. Если после проведения вычислений окажется, что выбранная величина приращения слишком велика, нам придется повторить вычисления с меньшим значением Δx.

Результат вычислений можно представить в виде табл. 2.1. В точке $x = 1$ определяется наклон касательной $g(x) = 2x = 2$. Значение функции y в конечной точке отрезка (y_1) вычисляется из значения функции y в начальной точке отрезка (y_0) по формуле

$$y_1 = y_0 + \Delta x\,(наклон) = 1 + 0{,}1(2) = 1.2. \qquad (2.6)$$

Полученное значение y_1 переносится во вторую строку табл. 2.1 и процесс повторяется. Проделав вычисления, получим $y = 3.90$ в точке $x = 2$. Отклонение от точного решения $y = x^2 = 4$ составляет 2,5%. Убедитесь в том, что выбор меньшего значения приращения Δx повышает точность решения и составьте новую табл. 2.1, используя значение шага $\Delta x = 0.05$.

ТАБЛИЦА 2.1. Итерационное решение дифференциального уравнения $dy/dx = 2x$ с начальным условием $y = 1$ при $x = 1$. Шаг $\Delta x = 0.1$. Показаны три значащие цифры

x	y	$g(x) = 2x$	$y_{n-1} + 0.1$ (наклон)
1.00	1.00	2.00	1.00 + 0.10(2.00) = 1.20
1.10	1.20	2.10	1.20 + 0.10(2.20) = 1.42
1.20	1.42	2.40	1.42 + 0.10(2.40) = 1.66
1.30	1.66	2.60	1.66 + 0.10(2.60) = 1.92
1.40	1.92	2.80	1.92 + 0.10(2.80) = 2.20
1.50	2.20	3.00	2.20 + 0.10(3.00) = 2.50
1.60	2.50	3.20	2.50 + 0.10(3.20) = 2.82
1.70	2.82	3.40	2.82 + 0.10(3.40) = 3.16
1.80	3.16	3.60	3.16 + 0.10(3.60) = 3.52
1.90	3.52	3.80	3.52 + 0.10(3.80) = 3.90
2.00	3.90		

2.4. ПРОГРАММА ДЛЯ КОМПЬЮТЕРА

Теперь, когда с помощью метода Эйлера получено численное решение дифференциального уравнения, мы в состоянии сформулировать этот метод в виде *алгоритма* для компьютера, т.е. конечной последовательности четких шагов или правил, которая решает задачу. Затем мы будем разрабатывать программу, реализующую этот алгоритм. Метод Эйлера описывается следующим алгоритмом:

1. Выбираются начальные условия, величина шага и количество итераций.
2. Определяется y и наклон в начальной точке отрезка.
3. Вычисляется значение y в конечной точке отрезка и печатается результат.
4. Шаги 2 и 3 повторяются требуемое число раз.

Приступая к разработке программы, первым делом необходимо раз-

бить всю задачу на последовательность независимых заданий, соответствующих вышеприведенным шагам. Например, можно записать основную программу в виде двух заданий:

```
CALL initial
CALL Euler
```

В языках Фортран и True BASIC эти задания записываются в виде последовательности инструкций *вызова* (call) *подпрограмм*; в языке Паскаль эти задания записываются указанием имен *процедур*.

Затем применим язык True BASIC для реализации каждого задания. Для простоты рассмотрим решение того же самого дифференциального уравнения $dy/dx = 2x$, о котором говорилось в разд. 2.3. Начальные условия и численные параметры задаются в подпрограмме **initial**.

```
SUB initial(y, x, dx, n)
  LET x = 1                    ! начальное значение x
  LET xmax = 2                 ! максимальное значение x
  LET y = 1                    ! начальное значение y
  LET dx = 0.1                 ! величина шага
  LET n = (xmax - x)/dx
END SUB
```

При написании данной программы использованы следующие элементы языка True BASIC:

1. Подпрограмма описывается инструкцией **SUB**.
2. Присваивание значений переменным осуществляется инструкцией **LET**.
3. Комментарии начинаются символом ! и могут располагаться в любом месте программы. Используйте их свободно, чтобы сделать свои программы читабельными. Компьютер их игнорирует.
4. Конец подпрограммы обозначается инструкцией **END SUB**.

Следующая наша задача заключается в реализации метода Эйлера и выводе на экран результата каждой итерации:

```
SUB Euler(y, x, dx, n)
 FOR i = 1 to n                ! число итераций n
   LET slope = 2*x             ! наклон в начальной точке отрезка
   LET change = slope*dx       ! вычисление полного изменения на отрезке
   LET y = y + change          ! новое значение y
   LET x = x + dx              ! приращение величины x
   PRINT x, y
 NEXT i
END SUB
```

В подпрограмме **Euler** использованы две дополнительные инструкции:

1. Цикл **FOR ... NEXT**. Управляющей переменной цикла является переменная *i* целого типа. Начальное значение устанавливается равным 1. После каждого прохождения цикла значение *i* увеличивается на 1. Цикл выполняется до тех пор, пока *i* не превысит *n*.
2. Инструкция **PRINT** выдает значение указанных переменных на экран.

Полный листинг программы приводится ниже:

```
PROGRAM example               ! начало основной программы
CALL initial(y, x, dx, n)
CALL Euler(y, x, dx, n)
END                           ! конец основной программы

SUB initial(y, x, dx, n)
 LET x = 1                    ! начальное значение x
 LET xmax = 2                 ! максимальное значение x
 LET y = 1                    ! начальное значение y
 LET dx = 0.1                 ! величина шага
 LET n = (xmax - x)/dx
END SUB

SUB Euler(y, x, dx, n)
 FOR i = 1 to n               ! число итераций n
   LET slope = 2*x            ! наклон в начальной точке отрезка
   LET change = slope*dx      ! вычисление изменения функции на отрезке
   LET y = y + change         ! новое значение y
   LET x = x + dx             ! приращение величины x
   PRINT x, y
 NEXT i
END SUB
```

Если вы не знакомы с языком программирования, запустите эту программу и посмотрите, как она работает. Отметим следующие особенности языка True BASIC, которые использованы в программе **example**:

1. Заголовок в первой строке основной программы необязателен.
2. Последняя строка основной программы должна содержать инструкцию **END**.
3. *Внешние* подпрограммы **initial**, **Euler** располагаются после инструкции **END** основной программы. Они представляют собой отдельные программные единицы и не имеют с основной программой общих переменных, если эти переменные не передаются в основную или из основной программы. В обеих подпрограммах передаются переменные *y*, *x*, *dx*, *n* из основной программы. В дальнейшем будут еще примеры использования внешних подпрограмм.
4. Выразительность программе придает распечатка ключевых слов заглавными буквами и выделение абзацами содержимого подпрограмм и циклов. Язык True BASIC не различает верхний и нижний регистры и игнорирует лишние пробелы.

Программа **Example** является примером *модульной* программы, т.е. программы, разбитой на отдельные задания, каждое из которых можно написать и проверить по отдельности. Всякая полная программа, включает по крайней мере *головную* программу, содержащую выполняемые инструкции. Обычно основная программа состоит из ряда вызовов или *обращений* к подпрограммам. Эти подпрограммы бывают двух видов: собственно *подпрограммы* (*subroutine*) и *функции* (*function*). Пример функций приводится в разд. 2.7.

Всегда, когда это возможно, мы будем пользоваться теми средствами языков True BASIC, Фортран и Паскаль, которые подчеркивают сходство всех трех языков. Поэтому в True BASIC мы будем использовать только *внешние* подпрограммы и функции. Внешние программные единицы описываются в любом порядке *после* инструкции **END** основной программы. Чтобы понять суть внешних подпрограмм, необходимо различать *локальные* и *глобальные* переменные. Имя переменной представляет ячейку памяти в компьютере. Внешняя подпрограмма является отдельной программной единицей, содержащей собственные *локальные* переменные. В том случае, когда используются одинаковые имена переменной в двух программных единицах, этим переменным отвечают две различные ячейки памяти. Например, в программе **local** имена переменных *x* и *y* используются в основной программе и подпрограмме **add**.

```
PROGRAM local
LET x = 1
LET y = 1
CALL add
PRINT x,y
END

SUB add
 LET x = x + 1
 LET y = y + 2
END SUB
```

Ввиду того что переменная *x — локальная*, на печать в основной программе будет выдано для *x* число 1. Тот же результат получится для переменной *y*.

Кроме того, необходимо описать *глобальные* переменные, которые используются в двух и более программных единицах. В языке True BASIC подпрограммы передают информацию в основную программу или в другие подпрограммы посредством параметров, содержащихся в обращении к подпрограмме. Передача информации осуществляется в обе стороны. Так, если переменная передается из основной программы в подпрограмму и в ней значение переменной изменяется, то обратно в головную программу она возвращается с новым значением. В программе **global1** иллюстрируется передача имен переменных *x* и *y*.

```
PROGRAM global1
LET x = 1
LET y = 1
CALL add(x,y)
PRINT x,y
END

SUB add(x,y)
 LET x = x + 1
 LET y = y + 2
END SUB
```

Какие значения переменных *x* и *y* напечатаются?

Говорить о том, что в подпрограмму передаются имена переменных, не совсем корректно. Правильнее сказать, что передаются не имена

переменных, а адреса ячеек памяти. Имя переменной — это просто метка, которой соответствует определенная ячейка памяти. Какие значения принимают *x* и *y* после выполнения программы **global2**, если подпрограмма **add** выглядит следующим образом:

```
PROGRAM global2
LET x = 1
LET y = 1
CALL add(x,y)
PRINT x,y
END

SUB add(r,s)
 LET r = r + 1
 LET s = s + 2
END SUB
```

Поэтому, мы должны помнить, что порядок следования и количество параметров в каждом обращении к подпрограмме и в ее описании должны совпадать. Какие значения принимают *x* и *y* после выполнения программы **global3**, если подпрограмма **add** выглядит следующим образом:

```
PROGRAM global3
LET x = 1
LET y = 1
CALL add(x,y)
PRINT x,y
END

SUB add(y,x)
 LET x = x + 1
 LET y = y + 2
END SUB
```

Локальные и глобальные переменные имеются и в Паскале, но описываются несколько иначе. В противоположность этому в стандартном Бейсике нет настоящих подпрограмм.

Почему уделяется такое внимание подпрограммам? Одно из важных практических соображений заключается в том, что их использование упрощает программирование. Если программу представлять себе в виде

последовательности заданий, то многие задания постоянно встречаются в разных алгоритмах. Поэтому можно использовать ранее написанные подпрограммы либо свои, либо чужие. Поскольку они являются самостоятельными программами, нет необходимости вычитывать их листинги и отслеживать использование переменных с одинаковыми именами в разных частях программы. Другая важная причина заключается в том, что использование подпрограмм способствует созданию модульных программ, в которых программные модули по возможности независимы друг от друга, так что любое изменение, вносимое в одну часть программы, не влияет на другие части. Такое разделение программы на части позволяет сконцентрировать усилия на одном модуле, облегчает проверку логики программы и обеспечивает надлежащее функционирование каждой части.

2.5. ПРОГРАММА ДЛЯ РЕШЕНИЯ ЗАДАЧИ ОБ ОСТЫВАНИИ КОФЕ

Вернемся к нашей чашечке кофе и разработаем компьютерную программу для численного решения уравнения теплопроводности Ньютона. Чтобы определить, разумна ли наша модель понижения температуры, приведем экспериментальные данные для остывания настоящей чашки кофе (табл. 2.2).

ТАБЛИЦА 2.2. Остывание чашечки кофе, помещенной в керамический стакан. Температура регистрировалась с точностью 0.1°C. Температура окружающего воздуха равнялась 22.0°C

Время, мин	T, °C	Время, мин	T, °C
0	83.0	8.0	64.7
1.0	77.7	9.0	63.4
2.0	75.1	10.0	62.1
3.0	73.0	11.0	61.0
4.0	71.1	12.0	59.9
5.0	69.4	13.0	58.7
6.0	67.8	14.0	57.8
7.0	66.4	15.0	56.6

Структура программы **cool**, в которой реализован алгоритм Эйлера численного решения задачи ньютоновой теплопроводности, аналогична структуре программы **example**. Единственное различие состоит в том, что вывод на экран осуществляется подпрограммой **output**, которая вызывается из подпрограммы **Euler**. Поскольку подпрограмма **output** со-

держит всего одну строку, может показаться, что в добавлении еще одной подпрограммы нет необходимости. Однако наш модульный подход к программированию будет полезен, когда в дальнейшем нам захочется выводить графики, а не печатать результаты.

```
PROGRAM cool          ! Метод Эйлера для задачи об остывании кофе
CALL initial(t, temperature, room_temp, r, dt, ncalc)
CALL Euler(t, temperature, room_temp, r, dt, ncalc)
END

SUB initial(t, temperature, room_temp, r, dt, ncalc)
 LET t = 0                    ! начальное время
 LET temperature = 83         ! начальная температура кофе (C)
 LET room_temp = 22           ! комнатная температура (C)
 LET r = 0.1                  ! коэффициент остывания (1/мин)
 LET dt = 0.1                 ! шаг по времени (мин)
 LET tmax = 2                 ! длительность (мин)
 LET ncalc = tmax/dt          ! общее количество шагов
END SUB

SUB Euler(t, temperature, room_temp, r, dt, ncalc)
 FOR icalc = 1 to ncalc       ! изменение вычисляется от начала интервала
  LET change = -r*(temperature - room_temp)
  LET temperature = temperature + change*dt
  LET t = t + dt              ! приращение времени
  CALL output(t, temperature)
 NEXT icalc
END SUB
SUB output(t, temperature)
 PRINT t, temperature         ! печать результатов
END SUB
```

ЗАДАЧА 2.1. Программа для решения задачи об остывании кофе

а. После того как мы набрали текст программы и устранили все синтаксические ошибки, как нам узнать, правильно ли в программе реализуется требуемый алгоритм? Например, может быть вы нечаянно вместо знака плюс набрали минус. Самое простое, что можно сделать, — это сравнить численные результаты с предельными случаями, для которых имеется аналитическое решение, или вычисления можно

проделать вручную. Используйте метод Эйлера и калькулятор для численного решения задачи теплопроводности Ньютона с теми же параметрами, что и в программе **cool**. Сравните свои расчеты с тем, что получилось по программе **cool**, и *проверьте* свою программу в этом случае.

б. Модифицируйте программу **cool**, используя инструкцию **INPUT PROMPT** (см. описание языка True BASIC), с тем чтобы значения параметров *r*, *dt*, *tmax* можно было вводить с клавиатуры.

в. Модифицируйте программу **cool** так, чтобы перед таблицей, содержащей время и температуру, печатался заголовок.

г. Используйте «вложенный» цикл **FOR NEXT** так, чтобы полученные результаты печатались не после каждого шага по времени, а в заданные моменты времени.

Поскольку вышеприведенная задача программирования может оказаться для вас новой, ниже приводится «решение». Заметим, что в модифицированной подпрограмме **Euler** производится *ncalc* итераций между обращениями к подпрограмме **output**. Конечно же, наше решение не единственное и должно рассматриваться только в качестве одного из вариантов.

```
PROGRAM cooler                          ! модифицированная программа
CALL initial(t, temperature, room_temp, r, dt, ncalc, nprt)
CALL output(t, temperature)             ! печать начальных значений
FOR iprt = 1 to nprt
  CALL Euler(t, temperature, room_temp, r, dt, ncalc)
  CALL output(t, temperature)           ! печать результатов
NEXT iprt
END
```

```
SUB initial(t, temperature, room_temp, r, dt, ncalc, nprt)
 LET t = 0
 LET temperature = 83                ! начальная температура кофе (C)
 LET room_temp = 22                  ! комнатная температура (C)
 INPUT prompt "коэффициент остывания r=":r ! коэффициент остывания
 INPUT prompt "длительность = ": tmax
 INPUT prompt "шаг по времени dt = ": dt
 LET print_period = 0.5              ! интервал (мин) вывода на печать
 LET nprt = tmax/print_period        ! число обращений к выводу результатов
 LET ncalc = print_period/dt         ! число итераций между печатями
 PRINT "время","температура"
 PRINT                               ! пропуск строки
END SUB

SUB Euler(t, temperature, room_temp, r, dt, ncalc)
 FOR icalc = 1 to ncalc
  LET change = -r*(temperature - room_temp)
  LET temperature = temperature + change*dt
 NEXT icalc
 LET t = t + dt*ncalc
END SUB

SUB output(t, temperature)
 PRINT t, temperature                ! печать результатов
END SUB
```

ЗАДАЧА 2.2. Анализ данных

а. Поскольку в качестве единицы измерения времени мы выбрали минуту, размерностью коэффициента остывания r будет мин$^{-1}$. Вы могли бы заметить, что в результате использования в программе cool значения $r = 0.1$ мин$^{-1}$ получается кривая охлаждения $T(t)$, которая не соответствует данным, приведенным в табл. 2.2. Используйте различные значения константы r для нахождения приближенного значения, которое соответствует «реальным» данным, приведенным в табл. 2.2. Убедитесь в том, что выбранное вами значение Δt достаточно мало и не оказывает влияния на получающуюся у вас зависимость температуры от времени. Зададимся следующими вопросами. Является ли ваше значение величины r правдоподобным? Применим ли закон теплопроводности Ньютона к чашечке кофе? В тех случаях,

когда коэффициент r гораздо больше или меньше единицы, что это говорит о нашем выборе единиц измерения времени? Увеличилось бы или уменьшилось значение коэффициента остывания r, если бы чашка была со специальной теплоизоляцией?

б. Начальная разность температур между кофе и окружающей средой равна 61°C. Сколько времени надо остужать кофе, чтобы эта разность температур составила 61/2 = 30.5°C? Через какое время разность температур уменьшится до 61/4 и до 61/8? Прежде чем проводить вычисления на компьютере, попытайтесь из простых соображений предугадать свои результаты.

в. Используйте значение коэффициента r, найденное в п. «а», и постройте график зависимости температуры от времени. Нанесите на тот же график данные из табл. 2.2 и сравните свои результаты. Хотя в разд. 2.7 мы научимся писать программы для построения графиков, неплохо будет предварительно построить графики вручную, чтобы «прочувствовать» свои данные.

г. Предположим, что начальная температура кофе 90°C, однако наслаждаться кофе можно, когда температура опуститься ниже 75°C. Допустим, что при 90°C добавление молока понижает температуру кофе на 5°C. Если вы торопитесь и хотите охладить кофе как можно быстрее, будете ли вы добавлять сначала молоко и ждать, пока кофе остынет, или же подождете до тех пор, пока кофе остынет до 80°C, а затем добавите молоко? Несмотря на то что вы, возможно, уже знаете ответ, используйте свою подпрограмму для «моделирования» этих двух случаев. Выберите значение коэффициента r, соответствующее реальной чашке кофе. Такой способ моделирования «что будет, если» применительно к «динамическим системам» часто используется в стратегических исследованиях (см., например, Робертс и др.).

2.6. УСТОЙЧИВОСТЬ И ТОЧНОСТЬ

Теперь, когда мы изучили применение метода Эйлера для численного решения дифференциального уравнения первого порядка, необходимо сформулировать некоторые практические рекомендации, которые позволят оценить точность метода. Поскольку мы заменили дифференциальное

уравнение его разностным аналогом, то естественно, что численное решение не может точно совпадать с «истинным» решением исходного дифференциального уравнения. В общем случае отклонение от точного решения обусловлено двумя причинами. Компьютеры не оперируют с вещественными числами (например, десятичными числами с дробной частью) бесконечой точности, а представляют числа с некоторым конечным числом десятичных цифр, определяемым аппаратными средствами компьютера или программным обеспечением. Арифметические операции, такие, как сложение или деление, оперирующие с вещественными числами, могут приводить к дополнительной погрешности, которая называется *погрешностью округления*. Например, если бы у нас был компьютер, оперирующий с вещественными числами, содержащими только две значащие цифры, то результатом умножения 2.1×3.2 было бы число 6.7. Важность погрешностей округления заключается в том, что они накапливаются по мере роста объема вычислений. В принципе мы выбирали алгоритмы, в которых погрешность округления заметным образом не накапливается, например мы старались не использовать вычитание чисел одного порядка.

Другой источник отклонения численного решения от точного обусловлен выбором алгоритма, а не точностью компьютера. В некоторых книгах по численному анализу такая погрешность называется *погрешностью приближения*. Поскольку эти погрешности зависят от выбора алгоритма, необходимо глубже изучить численный анализ и оценки погрешностей приближения. Тем не менее не существует никакого общего рецепта для выбора «наилучшего» метода численного решения дифференциальных уравнений. В последующих главах мы убедимся в том, что у каждого метода имеются свои достоинства и недостатки, а надлежащий выбор определяется вашими требованиями и характером конкретного решения, который может быть заранее не известен. Насколько точным должен быть ответ? На каком интервале требуется получить решение задачи? Какой компьютер имеется у вас в наличии? Сколько потребуется машинного и личного времени для решения задачи?

Практически точность численного решения определяют, уменьшая величину шага Δt до тех пор, пока численное решение не перестанет зависеть от шага при требуемом уровне точности. Само собой разумеется, необходимо осторожно выбирать величину шага, так как в случае очень малого Δt слишком сильно увеличивается число шагов, а значит, возрастают машинное время и погрешность округления.

Наряду с точностью алгоритма другой важной характеристикой пред-

ставляется его устойчивость. Например, может случиться так, что численные результаты находятся в хорошем соответствии с «истинным» решением на малых временах, а на больших временах отклоняются от него. Такое отклонение может происходить из-за того, что малые погрешности в алгоритме, многократно перемножаясь, приводят к геометрическому росту погрешности. Для конкретных задач такой алгоритм называется *неустойчивым*. В следующей задаче обсуждаются точность и устойчивость метода Эйлера.

ЗАДАЧА 2.3. Точность и устойчивость метода Эйлера

Для изучения точности метода Эйлера можно воспользоваться аналитическим решением дифференциального уравнения (2.1). Оно записывается в виде

$$T(t) = T_s - (T_s - T_0)\, e^{-rt}. \tag{2.7}$$

Заметим, что $T(t=0) = T_s - (T_s - T_0) = T_0$, а $T(t \longrightarrow \infty) = T_s$.

а. С помощью программы **cool** вычислите температуру в момент $t = 1$ мин с шагами $\Delta t = 0.1$, 0.05, 0.025, 0.01 и 0.005. Выберите значение коэффициента r, соответствующее реальному процессу. Постройте таблицу, содержащую разность между точным и численным решениями уравнения (2.7) как функцию Δt. Будет ли эта разность убывающей функцией Δt? Если шаг уменьшить в два раза, как изменится разность? Нарисуйте график разности как функцию Δt. В случае, когда точки приблизительно расположены на убывающей прямой, разность пропорциональна Δt (при $\Delta t \ll 1$). Если разность между аналитическим и численным решениями при заданном значении t пропорциональна $(\Delta t)^n$, численный метод называется методом n-го порядка точности. Какой порядок точности у метода Эйлера?

б. Какой необходимо выбрать величину шага Δt, чтобы достигалась точность 0.1% в момент времени $t = 1$? Какой необходимо выбрать величину шага Δt, чтобы достигалась точность 0.1% в момент времени $t = 5$?

в. Один из методов определения точности численного решения заключается в повторении вычислений с меньшим шагом и сравнении результатов. Если в обоих результатах совпадают n десятичных

цифр, то можно предположить, что будут совпадать и большее число десятичных знаков. Обсудим дифференциальное уравнение

$$R\frac{dQ}{dt} = V - \frac{Q}{C} \tag{2.8}$$

с начальным условием $Q = 0$ в момент времени $t = 0$. Это уравнение описывает зарядку конденсатора в RC-цепи с приложенным напряжением V. Время t измеряется в секундах и выбираются следующие характеристики цепи $R = 2000$ Ом, $C = 10^{-6}$ Ф и $V = 10$ В. Будет ли увеличиваться $Q(t)$ с течением времени? Увеличивается ли заряд Q до бесконечного значения или происходит насыщение? Напишите программу для численнного решения уравнения (2.8) методом Эйлера. Какой необходимо выбрать величину шага Δt, чтобы получить решение с тремя правильными десятичными знаками в момент времени $t = 0.005$ с?

г. Каковы особенности численного решения уравнения (2.8) для значений шагов $\Delta t = 0.005$, 0.004, 0.003? Приводит ли малое изменение шага Δt к большому изменению вычисляемой величины Q? Устойчив ли метод Эйлера для любой величины шага Δt?

2.7. ПРОСТЕЙШАЯ ГРАФИКА

Одно из преимуществ микрокомпьютеров заключается в способности легко и быстро строить графики. Несмотря на то что до сих пор не создан стандартный графический язык, основные графические инструкции на целом ряде типов компьютеров аналогичны, а понимание принципов работы простых стандартных графических программ на одном компьютере будет достаточным для того, чтобы освоить компьютер другого типа или легко одолеть другой графический язык. В дальнейшем мы рассмотрим основные графические команды языка True BASIC и разработаем программу для построения на экране компьютера графика любой функции. Основные графические команды для некоторых других языков и графических пакетов приводятся в конце первой части в приложении по языкам Фортран и Паскаль.

Графический монитор покрыт сеткой *«элементов изображения»* (*пикселов*). Количество пикселов зависит от аппаратной реализации. Одно из преимуществ языка True BASIC заключается в том, что общее число пикселов на экране монитора не имеет значения. Иначе говоря, отоб-

ражение абсолютных значений координат в координаты устройства, или пикселы, осуществляется в самом языке True BASIC. Первым делом при использовании графического вывода в языке True BASIC необходимо задать диапазоны изменения координат, в которых будут строиться графики. Инструкция

```
SET window xmin, xmax, ymin, ymax
```

определяет минимальное и максимальное значения x (горизонтальной) и y (вертикальной) координат и очищает экран. Инструкция

```
PLOT POINTS: x, y;
```

строит точку (x,y) в текущих оконных координатах. Несколько *«базовых»* графических инструкций языка True BASIC приводятся в табл. 2.3. Некоторые другие команды будут рассматриваться в следующих главах по мере необходимости.

ТАБЛИЦА 2.3. «Базовые» графические операторы языка True BASIC

```
PLOT POINTS: x, y
PLOT LINES: x1, y1; x2, y2;
PLOT TEXT, at x, y: expr$
BOX LINES: xmin, xmax, ymin, ymax
BOX ELLIPSE xmin, xmax, ymin, ymax
BOX AREA xmin, xmax, ymin, ymax
BOX CLEAR xmin, xmax, ymin, ymax
SET window xmin, xmax, ymin, ymax
SET COLOR color$
SET BACK color$
SET CURSOR line, column
CLEAR
FLOOD x, y
```

Запустите следующую программу и посмотрите, как работают графические команды. Для этого вы могли бы ввести в программу несколько инструкций **PAUSE**.

```
PROGRAM graphics
SET window 0,100,0,100
PLOT TEXT, at 35,90: "демонстрационная программа"
LET xmin = 10
LET ymin = 10
LET xmax = 30
LET ymax = 30
FOR i = 1 to ymin
  PLOT POINTS: i, xmin
NEXT i
PAUSE 1                          ! пауза длительностью 1 с
PLOT LINES: xmin, xmax, ymin, ymax
BOX LINES: xmin, xmax, ymin, ymax
BOX ELLIPSE xmin, xmax, ymin, ymax
BOX AREA xmin, xmax, ymin, ymax
BOX CLEAR xmin, xmax, ymin, ymax
SET back "black"
CLEAR
SET color "white"
SET cursor 10,5    ! координаты указываются в знакоместах, а не оконные
PRINT "демонстрационная программа"
BOX AREA xmin, xmax, ymin, ymax
BOX LINES 50,60,5,60
END
```

Теперь мы воспользуемся некоторыми графическими командами и создадим программу, которая строит на экране любую функцию. Структура основной программы такова:

```
PROGRAM plot                     ! пример программы построения графика
CALL minmax(xmin,xmax,ymin,ymax,title$)
CALL plot_axis(xmin,xmax,ymin,ymax,title$)
CALL plot_function(xmin,xmax)
END
```

Минимальное и максимальное значения x и y вводятся с клавиатуры в подпрограмме minmax.

```
SUB minmax(xmin, xmax, ymin, ymax, title$)
  INPUT prompt "минимум горизонтальной переменной? ":xmin
  INPUT prompt "максимум горизонтальной переменной? ":xmax
  INPUT prompt "минимум веритикальной переменной? ":ymin
  INPUT prompt "максимум вертикальной переменной? ":ymax
  INPUT prompt "название графика? ":title$
END SUB
```

Подпрограмма **plot_axis** строит вертикальную и горизонтальную оси координат, размечает их и надписывает оси и график. Оптимальное количество и расположение делений и надписей зависят от разрешающей способности экрана. Числовые значения, соответствующие каждому делению, не наносятся, поскольку размер литер может не соответствовать масштабу графика. Обратите внимание на использование инструкции **PLOT TEXT** с функцией **PRINT using$** и управляющей структурой **IF–THEN–ELSE**.

```
SUB plot_axis(xmin, xmax, ymin, ymax, title$)
  LET ntick = 10                          ! количество делений на осях
  ! расстояние между делениями на оси x
  LET dx = (xmax - xmin)/ntick
  ! расстояние между делениями на оси y
  LET dy = (ymax - ymin)/ntick
  ! определение мировых или оконных координат
  SET window xmin - dx, xmax + dx, ymin - dy, ymax + dy
  ! определения расположения осей
  IF xmin*xmax < 0 then
     LET x0 = 0
  ELSE
     LET x0 = xmin
  END IF
  IF ymin*ymax < 0 then
     LET y0 = 0
  ELSE
     LET y0 = ymin
  END IF
  ! построение осей
  PLOT LINES: x0, ymin; x0, ymax          ! вертикальная ось
  PLOT LINES: xmin, y0; xmax, y0          ! горизонтальная ось
```

```
! определение длины деления
LET Lx = 0.1*dy                          ! шаг между делениями на оси x
LET Ly = 0.1*dx                          ! шаг между делениями на оси y
! нанесение делений
FOR itick = 0 to ntick
   LET col = xmin + itick*dx
   LET row = ymin + itick*dy
   PLOT LINES: col, y0 - Lx; col, y0 + Lx
   PLOT LINES: x0 - Ly, row; Ly + x0, row
NEXT itick
! печать названия графика
PLOT TEXT, at xmin + 7*dx, ymax: title$
! отметка максимального и минимального значений x и y
PLOT TEXT, at xmax - 0.5*Lx, y0: using$("***.*", xmax)
PLOT TEXT, at x0 - 4*Ly, ymax + Ly: using$("***.*",ymax)
END SUB
```

Инструкция **PLOT** используется в **подпрограмме plot_function** для построения графика функции $T(t)$, определяемой соотношением (2.7). Поскольку $T(t)$ описана как *внешняя* функция, в каждой подпрограмме, которая ее использует, необходимо наличие инструкции **DECLARE DEF**

```
SUB plot_function(xmin, xmax)
 DECLARE DEF f    ! в подпрограмме используется внешняя функция f(t)
 FOR t=xmin to xmax step 0.01
  PLOT t,f(t);       ! сокращенная форма от PLOT LINES: t,f(t)
 NEXT t
END SUB
```

Функция $T(t)$ описывается в отдельной подпрограмме. Главное различие между функциями и подпрограммами заключается в том, что в подпрограммы-функции входит относительно небольшое количество параметров и возвращается единственное значение.

```
DEF f(t)                 ! начало описания
 LET r = 0.07
 LET TS = 22
 LET TO = 83
 LET f = TS + (TO - TS)*exp(-r*t)
END DEF                  ! конец описания
```

ЗАДАЧА 2.5. Время остывания

а. Включите вышеприведенные подпрограммы в программу **cooler** так, чтобы численное решение для температуры кофе как функции времени не выдавалось на печать, а строилось в виде графика. (Примером программы, которая строит график по числовым данным, является программа **Fall**, приведенная в гл. 3.)

б. Найдите время, необходимое для того, чтобы разность температур между температурой кофе и комнатной температурой составила $1/e \approx 0.37$ от начальной. Это время называется *временем релаксации* или *временем остывания*. Зависит ли время релаксации от начальной температуры кофе или комнатной температуры? Попробуйте взять различные значения коэффициента r и определите качественную зависимость времени релаксации от r.

2.8. ПЕРСПЕКТИВА

Хотя эта глава вводная, как могли почувствовать некоторые более подготовленные читатели, тем не менее вы познакомились со многими новыми понятиями и методами. Если вы не знакомы с программированием, то первые опыты с компьютером и введение в синтаксис нового языка могли вас слегка озадачить. Но наберитесь храбрости. Осталось изучить совсем немного синтаксических конструкций языка. Если вы заглянете в распечатки программ, встречающиеся в последующих главах, то, вероятно, узнаете большинство используемых синтаксических конструкций.

ЛИТЕРАТУРА

George B. Arfken, David F. Griffing, Donald C. Kelly, Joseph Priest, University Physics, Academic Press, 1984. Гл. 23 посвящена переносу тепла. В ней обсуждается закон теплопроводности Ньютона.

Nancy Roberts, David Andersen, Ralph Deal, et al., Introduction to Computer Simulations: The System Dynamics Approach, Addison-Wesley, 1983. Книга посвящена численному моделированию в науках об обществе. Численное решение системы нелинейных уравнений модели находится с помощью алгоритма Эйлера.

Не представляется возможным перечислить все прекрасные книги по численным методам, приведем здесь лишь ссылки на некоторые книги вводного характера.

Forman S. Acton, Numerical Methods That Work, Harper & Row, 1970. Книга рассчитана на подготовленного читателя, но написана понятным языком.

L. V. Atkinson, P. J. Harley, An introduction to Numerical Methods with Pascal, Adisson-Wesley, 1983. Книга предполагает хорошее знание читателем языка Паскаль и математического анализа, но не предполагает никакой подготовки по численным методам.

Samuel D. Conte, Carl de Boor, Elementary Numerical Analysis: an Algorithmic Approach, McGraw-Hill, 1972.

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

Гутер Р. С., Овчинский Б. В., Элементы численного анализа и математической обработки результатов опыта. — М.: Наука, 1970. Гл. 4 посвящена численному решению дифференциальных уравнений. В § 26 подробно обсуждается метод Эйлера и его погрешность. Изложение ведется на том же уровне, что и в данной книге.

Кунц К. С. Численный анализ. — Киев : Техніка, 1964. В гл. 8—9 подробно исследуются различные методы решения обыкновенных дифференциальных уравнений, их погрешность и точность. Книга рассчитана на подготовленного читателя.

ПАДЕНИЕ ТЕЛ

3

В этой главе мы познакомимся с методом численного решения уравнений движения Ньютона и обсудим количественные характеристики и качественное поведение тел, падающих вблизи поверхности Земли.

3.1. ОСНОВНЫЕ ПОНЯТИЯ

Типичным примером движения является падение предметов у земной поверхности. Простейшее описание такого движения не учитывает возможного вращения и внутренних движений и описывает некий идеализированный объект, который называется *материальной точкой*, т.е. объект, не обладающий внутренней структурой. Конечно же, такие объекты, как планеты, камни, бейсбольные мячи и атомы, не являются «точками». Тем не менее во многих случаях можно пренебречь их внутренним движением и считать их материальными точками.

Сначала мы рассмотрим одномерное движение, для описания которого нужна только одна пространственная координата. Известно, что мгновенные координату $y(t)$, скорость $v(t)$ и ускорение $a(t)$ материальной точки можно определить на языке дифференциальных уравнений:

$$v(t) = \frac{dy(t)}{dt} \tag{3.1}$$

и

$$a(t) = \frac{dv(t)}{dt}. \tag{3.2}$$

Эти величины называются *кинематическими*, поскольку они описывают движение безотносительно причины его вызывающей.

Зачем в кинематике необходимо понятие ускорения? Ответить на этот вопрос можно только *a posteriori*. Благодаря Ньютону известно, что ускорение материальной точки определяется действующей на нее результирующей силой. Второй закон движения Ньютона записывается следующим образом:

$$a(t) = \frac{1}{m} F(y,v,t), \tag{3.3}$$

где F — равнодействующая *сила*, m — *инертная масса*. В общем случае сила зависит от координаты, скорости и времени. Обратите внимание на скрытый смысл закона Ньютона, заключающийся в том, что движение материальной точки не зависит от d^2v/dt^2 и любых производных по скорости более высокого порядка. Тот факт, что мы можем найти простые объяснения для движения, является свойством природы, а не математического описания.

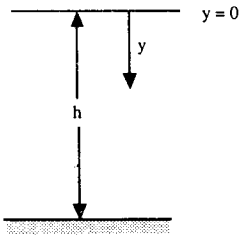

Рис. 3.1. Удобная система координат для задачи о теле, падающем с высоты h.

Для описания движения материальной точки необходимо решить систему двух дифференциальных уравнений первого порядка (3.1) и (3.2). Часто уравнения (3.1) и (3.2) объединяют в одно дифференциальное уравнение *второго порядка* относительно координаты:

$$\frac{d^2y(t)}{dt^2} = \frac{F}{m}. \tag{3.4}$$

3.2. СИЛА, ДЕЙСТВУЮЩАЯ НА ПАДАЮЩЕЕ ТЕЛО

В отсутствие сопротивления воздуха все тела независимо от их массы, размеров и состава на одинаковом расстоянии от земной поверхности имеют одинаковое ускорение. Такое идеализированное движение, при котором сопротивлением воздуха пренебрегают, называется «свободным падением». Ускорение свободно падающего тела обычно обозначают буквой g и направляют к земной поверхности. Вблизи поверхности Земли значение g приблизительно равно 9.8 м/с^2. Выберем систему координат с положительным направлением вниз, как показано на рис. 3.1. В этом случае $a = +g$ и решение уравнения (3.4) можно записать в виде

$$v(t) = v_0 + gt \tag{3.5а}$$

и

$$y(t) = y_0 + v_0t + \frac{1}{2}gt, \tag{3.5б}$$

где y_0 и v_0 обозначают начальные координату и скорость материальной точки соответственно. Заметим, что для точного определения движения необходимо два начальных условия.

Для свободного падения вблизи земной поверхности аналитическое решение уравнения (3.5) является настолько простым, что нет необходимости останавливаться на нем подробно. Однако нетрудно представить реалистические изменения уравнений движения в гравитационном поле Земли, которые не будут иметь простых аналитических решений. Например, ускорение не будет константой, если учитывать зависимость силы тяжести от расстояния до центра Земли. В соответствии с законом тяготения Ньютона сила, действующая на тело массой m, равна

$$F = \frac{GMm}{(R + y)^2} = \frac{gm}{(1 + y/R)^2}, \tag{3.6}$$

где y — расстояние от поверхности Земли, R — радиус Земли, G — постоянная всемирного тяготения, M — масса Земли, $g = GM/R^2$.

Другой важной модификацией задачи о свободном падении является учет тормозящей силы, обусловленной сопротивлением воздуха. Направление этой тормозящей силы должно быть противоположно скорости движения тела. Рассмотрим сначала падение материальной точки. Тормозящая сила F_d направлена вверх, как показано на рис. 3.2. Если воспользоваться системой координат, показанной на рис. 3.1, то пол-

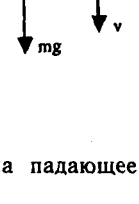

Рис. 3.2. Силы, действующие на падающее тело с учетом сопротивления воздуха.

ную силу, действующую на материальную точку, можно записать в виде

$$F = F_g - F_d = mg - F_d . \tag{3.7}$$

В общем случае зависимость F_d от скорости необходимо определять эмпирически, проводя конечную серию наблюдений положения тела. Один из способов определения функции $F_d(v)$ заключается в измерении коор-

динаты y как функции t и получении зависимости скорости и ускорения от времени. Используя эту информацию, можно найти зависимость ускорения от скорости v и получить функцию $F_d(v)$. Однако обычно такой метод непригоден, поскольку вычисление наклона кривых, необходимое для нахождения скорости и ускорения, сопряжено с большими ошибками. Более хороший метод представляет собой обратную процедуру. Для функции F_d предполагается какой-то определенный вид зависимости от скорости v, и эта формула используется для нахождения функции $y(t)$. Если вычисленные значения функции $y(t)$ согласуются с экспериментальными, то предложенная зависимость $F_d(v)$ считается экспериментально подтвержденной.

Наиболее общими зависимостями силы сопротивления от скорости являются

$$F_d(v) = k_1 v \tag{3.8а}$$

и

$$F_d(v) = k_2 v^2, \tag{3.8б}$$

а параметры k_1 и k_2 зависят от свойств среды и геометрии тела. Подчеркнем, что зависимости (3.8) не являются точными законами физики, а представляют собой полезные *феноменологические* выражения, приближенно описывающие $F_d(v)$ в ограниченном диапазоне скоростей. Ввиду того что функция $F_d(v)$ возрастающая, существует *предельная*, или *установившаяся*, скорость, соответствующая условию $F_d = F_g$ и нулевому ускорению. Эту скорость можно найти из (3.7) и (3.8). В результате получим

$$v_1 = \frac{mg}{k_1} \tag{3.9б}$$

или

$$v_2 = \left[\frac{mg}{k_2}\right]^{1/2} \tag{3.9б}$$

соответственно для линейного и квадратичного случаев. Часто удобно измерять скорости в единицах установившейся скорости. Используя (3.8) и (3.9), выпишем F_d для линейного и квадратичного случаев:

$$F_d = k_1 v_1 \left(\frac{v}{v_1}\right) = mg\left(\frac{v}{v_1}\right) \tag{3.10а}$$

и

$$F_d = k_2 v_2^2 \left(\frac{v}{v_2}\right)^2 = mg\left(\frac{v}{v_2}\right)^2. \tag{3.10б}$$

Отсюда равнодействующую силу, действующую на падающее тело, можно записать в виде

$$F_1(v) = mg\left(1 - \frac{v}{v_1}\right) \tag{3.11а}$$

или

$$F_2(v) = mg\left(1 - \frac{v^2}{v_2^2}\right). \tag{3.11б}$$

Чтобы понять, насколько существенно влияет сопротивление воздуха на падение обычных тел, рассмотрим движение камня массой $m = 10^{-2}$ кг. Установлено, что с хорошей степенью точности сила сопротивления пропорциональна v^2. Для камня радиусом 0.01 м коэффициент k_2, как экспериментально установлено, равен $k_2 \approx 10^{-4}$ кг/м. Из выражения (3.9б) найдем, что установившаяся скорость равна приблизительно 30 м/с. Поскольку эта скорость достигается свободно падающим телом, пролетевшим по вертикали приблизительно 50 м за 3 с, то можно ожидать, что на значительно меньших временах и расстояниях сопротивление воздуха играет существенную роль. Поэтому многие задачи, с которыми мы встречаемся в элементарных курсах механики, не являются реалистическими.

3.3. ЧИСЛЕННОЕ РЕШЕНИЕ УРАВНЕНИЙ ДВИЖЕНИЯ

Поскольку аналитически решить уравнения движения (3.4) с равнодействующей силой, определяемой выражением (3.11б), не просто, мы вынуждены прибегнуть к численным методам. Метод Эйлера просто обобщается на случай решения дифференциального уравнения второго порядка. Сначала переписываем уравнение (3.4) в виде системы двух дифференциальных уравнений первого порядка (3.1) и (3.2). Обозначим через Δt шаг по времени, тогда момент времени t_n, соответствующий n-му шагу, равен

$$t_n = t_0 + n\Delta t. \tag{3.12}$$

Обозначим также через a_n, v_n и y_n значения ускорения, скорости и координаты на n-м шаге, например $a_n = a_n(y_n, v_n, t_n)$. Прямое обобщение метода Эйлера, который рассматривался в гл. 2, принимает вид

$$v_{n+1} = v_n + a_n \Delta t \tag{3.13а}$$

и

$$y_{n+1} = y_n + v_n \Delta t. \tag{3.13б}$$

Заметим, что v_{n+1} — скорость в конечной точке интервала — вычисляется через a_n — производную скорости в *начальной точке* этого интервала. Аналогично y_{n+1} — координата в конечной точке интервала вычисляется через v_n — производную координаты в *начальной точке* интервала.

Использованный алгоритм численного решения дифференциального уравнения не является единственным. Например, одно простое изменение выражений (3.13) состоит в том, чтобы определять y_{n+1} через v_{n+1} — скорость в *конечной точке* интервала, а не в начальной точке. Запишем такой модифицированный метод Эйлера в следующем виде:

$$v_{n+1} = v_n + a_n \Delta t \tag{3.14а}$$

и

$$y_{n+1} = y_n + v_{n+1} \Delta t. \tag{3.14б}$$

Поскольку алгоритм (3.14), был тщательно рассмотрен Кромером (см. список литературы), мы будем называть выражения (3.14) методом Эйлера — Кромера. [Кромер называет выражения (3.13) и (3.14) аппроксимациями *по первой точке* и *по последней точке*.] Существует ли какое-нибудь соображение *a priori*, какому из двух методов отдать предпочтение?

3.4. ОДНОМЕРНОЕ ДВИЖЕНИЕ

В программе **fall** представлена простая реализация метода Эйлера применительно к задаче о движении свободно падающего тела. Структура этой программы похожа на структуру программы **cooler** с тем отличием, что здесь использована конструкция **DO...LOOP** с условием **WHILE**. Общие особенности циклов **DO** можно понять, запустив эту простую программу.

```
PROGRAM do_loop
DO while i < 10
  LET i = i + 1
  PRINT i
LOOP
PAUSE 1
LET i = 0
DO
  LET i = i + 1
  PRINT i
LOOP until i = 10
END
```

Условия **WHILE** или **UNTIL** можно использовать в конструкциях как **DO**, так и в **LOOP** или и в той, и в другой. Есть ли какая-нибудь разница между условиями **WHILE** и **UNTIL** в программе do_loop?

```
PROGRAM fall                          ! свободно падающее тело
CALL initial(y, v, t, g, dt, height)  ! начальные условия и параметры
CALL print_parameters(dt, ncalc)      ! ncalc число шагов между печатью
CALL print_table(y, v, g, t)          ! печать начальных значений
DO
  CALL Euler(y, v, accel, t, g, dt, ncalc)  ! решение разностного уравнения
  CALL print_table(y, v, accel, t)    ! печать результатов через ncalc шагов
LOOP UNTIL y > height
END

SUB initial(y, v, t, g, dt, height)
  LET t = 0                           ! начальный момент времени (с)
  LET y = 0                           ! начальное значение координаты (м)
  LET height = 10                     ! начальная высота тела над Землей
  LET v = 0                           ! начальная скорость
  INPUT prompt "шаг по времени dt = ": dt
  LET g = 9.8                         ! значение ускорения свободного падения
END SUB
```

В следующих ниже подпрограммах определяются некоторые параметры и печатается заголовок таблицы.

```
SUB print_parameters(dt,ncalc)
  LET print_period = 0.1          ! (c)
  LET ncalc = print_period/dt  ! число шагов между печатью результатов
  PRINT "время(с)", "y(м)", "скорость(м/с)", "ускорение(м/(с*с))"
  PRINT
END SUB

SUB print_table(y,v,accel,t)          ! печать результатов в виде таблицы
  PRINT t,y,v,accel
END SUB

SUB Euler(y,v,accel,t,g,dt,ncalc)
  FOR icalc = 1 to ncalc
    LET y = y + v*dt                  ! выбирается скорость в начальной точке
    LET accel = g                     ! ось y направлена вниз
    LET v = v + accel*dt
  NEXT icalc
  LET t = t + dt*ncalc
END SUB
```

Поучительно выяснить, каким образом можно изменить программу **fall**, чтобы выдавать результаты в различной форме. Например, простой способ «пронаблюдать» траекторию падающего тела заключается в замене подпрограмм **print_parameters** и **print_table** на соответствующие подпрограммы:

```
SUB plot_parameters(height,dt,ncalc)
  LET plot_period = 0.25
  LET ncalc = plot_period/dt
  SET window 0,20,height,-1
END SUB

SUB plot_trajectory(y)
  LET x = 10                          ! размещаем линию посредине экрана
  LET r = 0.1
  BOX AREA x - r,x + r,y - r,y + r  ! рисование частицы
END SUB
```

Если бы мы пожелали получить графики скорости и координаты падающего тела в зависимости от времени, то в этом случае можно было бы заменить подпрограммы печати соответственно на **graph_parameters** и **graph**.

```
SUB graph_parameters(height, t, dt, ncalc)
  LET plot_period = 0.05
  LET ncalc = plot_period/dt
  LET tmin = 1
  LET tmax = 2
  LET ymax = height
  LET ymin = 0
  LET title1$ = "свободно падающее тело"
  LET title2$ = "y как функция t"
  SET plot_axis(tmin, tmax, ymin, ymax, title1$, title2$)
END SUB

SUB graph(y, t)
  PLOT LINES: t, y;
END SUB
```

Обратите внимание на то, что подпрограмма **graph_parameters** вызывает подпрограмму **plot_axis**, которую мы разработали в гл. 2. Единственное изменение, которое нам может быть желательно сделать, — это передавать в подпрограмму названия двух осей, а не одной, как в подпрограмме **plot_axis**. Хотя нет надобности опять распечатывать подпрограмму **plot_axis**, мы сделаем это, чтобы не оставалось никаких вопросов.

```
SUB plot_axis(xmin, xmax, ymin, ymax, title$, title2$
  LET ntick = 10                    ! число делений
  LET dx = (xmax - xmin)/ntick      ! расстояние между делениями на оси x
  LET dy = (ymax - ymin)/ntick      ! расстояние между делениями на оси y
  SET window xmin - dx, xmax + dx, ymin - dy, ymax + dy
  IF xmin*xmax < 0 then
    LET x0 = 0
  ELSE
    LET x0 = xmin
  END IF
```

```
  IF ymin*ymax < 0 then
    LET y0 = 0
  ELSE
    LET y0 = ymin
  END IF
  PLOT LINES: x0,ymin; x0,ymax        ! вертикальная ось
  PLOT LINES: xmin,y0; xmax,y0        ! горизонтальная ось
  ! определяем длину делений
  LET Lx = 0.1*dy                     ! длина деления на оси x
  LET Ly = 0.1*dx                     ! длина горизонтального деления на оси y
  FOR itick = 0 to ntick
    LET col = xmin + itick*dx
    LET row = ymin + itick*dy
    PLOT LINES: col,y0 - Lx; col,y0 + Lx
    PLOT LINES: x0 - Ly,row; Ly + x0,row
  NEXT itick
  ! координаты x, y зависят от длины надписи
  PLOT TEXT, at xmin + 4*dx,ymax: title1$
  PLOT TEXT, at xmin + 4*dx,ymax - 1: title2$
  PLOT TEXT, at xmax - 0.5*Lx,y0: using$("###.#",xmax)
  PLOT TEXT, at x - 4*Ly,ymax + Ly: using$("###.#",ymax)
END SUB
```

В следующих четырех задачах мы воспользуемся простыми модификациями программы **fall** для исследования движения простых тел с учетом и без учета сопротивления воздуха.

ЗАДАЧА 3.1. Сравнение алгоритмов

а. Используйте программу **fall** для определения временной зависимости скорости и координаты свободно падающего тела вблизи земной поверхности. Выберите в качестве начальных условий $y = 0$, *height* = 10 и $v = 0$. (Параметр *height* — начальное расстояние начала координат от поверхности Земли.) Система координат, используемая в программе **fall**, показана на рис. 3.1. Каким будет подходящее значение шага Δt? Сравните полученные вами результаты с точными, вычисленными по формулам (3.5).

б. Измените программу **fall** и реализуйте метод Эйлера—Кромера. (Все, что необходимо, — это поменять местами две строки в подпрограмме **Euler**.) Существует ли какое-нибудь соображение, позволяющее отдать предпочтение одному из алгоритмов? Придумайте простое изменение любого из двух алгоритмов, чтобы для свободно падающего тела получались точные результаты (сопротивление воздуха не учитывать). Примените метод Эйлера или Эйлера—Кромера для решения остальных задач этой главы.

в. Используйте подпрограмму **trajectory** для слежения за координатой падающего тела через равные промежутки времени. Одинаково ли расстояние между положениями тела в разные моменты времени?

г. Для сравнения $y(t)$ и $v(t)$ с соответствующими функциями при учете сопротивления воздуха постройте с помощью подпрограммы **plot_graph** графики y, v и a как функции времени.

Теперь, после того как мы проверили наши программы в случае, когда не учитывается сопротивление воздуха, можно рассмотреть некоторые более реалистические задачи. В табл. 3.1 приводятся результаты измерений координат падающего пенопластового шарика в зависимости от времени. Необходимо ли учитывать влияние сопротивления воздуха?

ТАБЛИЦА 3.1. Результаты Гринвуда, Ханна и Милтона (см. список литературы) для случая вертикального падения пенопластового шарика массой 0.254 г и радиусом 2.54 см

t, с	Координата, м
−0.132	0.0
0.0	0.075
0.1	0.260
0.2	0.525
0.3	0.870
0.4	1.27
0.5	1.73
0.6	2.23
0.7	2.77
0.8	3.35

ЗАДАЧА 3.2. Падение пенопластового шарика

Модифицируйте программу **fall** так, чтобы равнодействующая сила задавалась либо формулой (3.11а), либо (3.11б). Для каждого вида силы определите значение установившейся скорости, при которой вычисленные значения функции $y(t)$ лучше всего согласуются с экспериментальными данными, приведенными в табл. 3.1. Нанесите оба набора вычисленных значений $y(t)$ и экспериментальные данные на один график и визуально определите, какая кривая $y(t)$ дает в целом лучшее согласие с экпериментом. Чем качественно различаются обе теоретические кривые $y(t)$? Указание: поскольку экспериментальные значения функции y, приведенные в табл. 3.1, вначале идут через неравные интервалы времени, то, возможно, вам захочется модифицировать свою программу следующим образом:

```
PROGRAM styrofoam
CALL initial(y, v, t, g, vt2, dt, height)
CALL print_parameters(t, dt, n0, ncalc)
CALL print_table(y, v, g, t)                    ! печать начальных условий
CALL Euler(y, v, accel, t, g, vt2, dt, n0)
CALL print_table(y, v, accel, t)                ! печать значений в момент t = 0
DO
  CALL Euler(y, v, accel, t, g, vt2, dt, ncalc)
  CALL print_table(y, v, accel, t)
LOOP UNTIL y > height
END

SUB initial(y, v, t, g, vt2, dt, height)
  LET t = -0.132                                ! начальный момент времени
  LET y = 0                                     ! начальное смещение (м)
  LET height = 4                                ! начальная высота тела над землей
  LET v = 0                                     ! начальная скорость
  INPUT prompt "шаг по времени = ": dt
  LET g = 9.8
  INPUT prompt "установившаяся скорость (м/с) = ": vterm
  LET vt2 = vterm*vterm
END SUB
```

```
SUB print_parameters(t, dt, n0, ncalc)
  LET print_period = 0.1                    ! (c)
  LET ncalc = print_period/dt
  LET n0 = -t/dt
  PRINT "время(с)","y(м)","скорость(м/с)","ускорение (м/(с*с))"
  PRINT
END SUB
```

Обратите внимание на то, что подпрограмма **Euler** вызывается из основной программы с двумя различными значениями числа шагов между последовательными выдачами результатов. Подпрограмма **Euler** не приводится, потому что она представляет собой просто обобщение подпрограммы с таким же названием из распечатки программы **fall**.

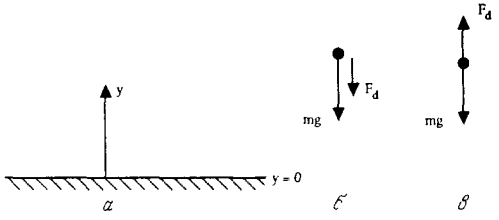

Рис. 3.3. Принятая в задаче 3.3 система координат с осью y, направленной вверх от земной поверхности (*а*). Силы, действующие на тело, движущееся вверх (*б*). Силы, действующие на тело, движущееся вниз (*в*).

ЗАДАЧА 3.3. Влияние сопротивления воздуха на движение камня вверх и вниз

а. Давайте проверим утверждение о заметном влиянии сопротивления воздуха на движение камня, сделанное в разд. 3.2. Вычислите скорость, с какой камень упадет на землю, если он падал с высоты 50 м из состояния покоя. Сравните эту скорость с той, которую приобретает свободно падающее тело при тех же условиях. Примите, что тормозящая сила пропорциональна v^2, а установившаяся скорость равна 30 м/с.

б. Предположим, что тело брошено вертикально вверх с начальной скоростью v_0. Известно, что если пренебречь сопротивлением воздуха, то максимальная высота, на которую поднимется тело, равна $v_0^2/2g$, скорость, с которой тело падает на земную поверхность, равна v_0, время подъема и время падения одинаковы, а общее время движения составляет v_0/g. Прежде чем проводить численное моделирование, приведите простое качественное объяснение того, как, по вашему мнению, повлияет сопротивление воздуха на эти величины. Затем проведите численные расчеты и проверьте правильность своих качественных объяснений. Предположите, что $F_d \sim v^2$, а установившаяся скорость равна 30 м/с. Указания: выберите систему координат, как на рис. 3.3, с положительным направлением оси y вверх. Чему равна равнодействующая сила при $v > 0$ и $v < 0$? Возможно, вы сочтете удобным воспользоваться функцией **sgn**, поскольку sgn(x) имеет своим значением знак x. Или можно записать тормозящую силу в виде $F_d = -v*\text{abs}(v)$. Один из способов определения максимальной высоты, на которую поднимется камень, заключается в использовании инструкции

```
IF v*vold < 0 then PRINT "максимальная высота = "; y
```

где $v = v_n$, а $vold = v_{n-1}$.

ЗАДАЧА 3.4. Сила, зависящая от координаты

Если сила зависит от координаты, как в формуле (3.6), то не существует простого аналитического решения для $y(t)$. Модифицируйте программу **fall**, чтобы промоделировать падение материальной точки под действием силы (3.6). Предположите, что материальная точка падает с высоты h с нулевой начальной скоростью, и вычислите ее скорость в момент удара о земную поверхность. Определите значение h, для которого эта скорость соударения отличается на один процент от скорости, приобретаемой при движении с постоянным ускорением $g = 9.8$ м/с^2. Радиус Земли примите равным $6.37 \cdot 10^6$ м.

3.5. ДВУМЕРНЫЕ ТРАЕКТОРИИ

По всей видимости, вам известны двумерные задачи, в которых пренебрегается сопротивлением воздуха. Например, если бросить в воздух мяч с начальной скоростью v_0 под углом θ_0 к горизонту, то как далеко улетит мяч в горизонтальном направлении, на какую максимальную высоту он поднимется и каково будет время полета? Предположим, что мяч брошен на ненулевой высоте h от земной поверхности. Под каким углом его следует бросить, чтобы он упал на максимальном расстоянии от точки бросания? Будут ли верны ваши ответы, если учесть сопротивление воздуха? Такого рода вопросы мы и обсудим ниже.

Рассмотрим тело массой m с начальной скоростью $\mathbf{v}_0$, направленной под углом θ_0 к горизонту (рис. 3.4,*а*). На материальную точку действуют сила тяжести и тормозящая сила, равные соответственно mg и F_d, а направление последней противоположно скорости $\mathbf{v}$ (рис. 3.4,*б*). Запишем уравнения движения Ньютона для компонент x и y:

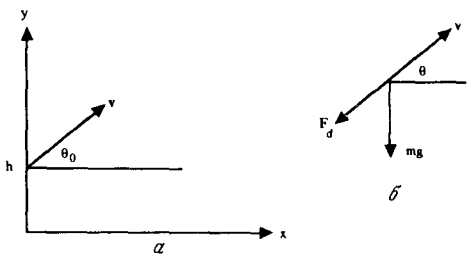

Рис. 3.4. Мяч бросают с высоты h под углом θ_0 к горизонту с начальной скоростью $\mathbf{v}_0$(*а*). Сила тяжести и тормозящая сила, действующие на материальную точку (*б*).

$$m\frac{dv_x}{dt} = -F_d\cos\theta, \tag{3.15а}$$

$$m\frac{dv_y}{dt} = -mg - F_d\sin\theta. \tag{3.15б}$$

В качестве примера определим максимальное расстояние от точки бросания, на которое улетит круглый стальной шар радиусом ≈ 4 см. Разумно предположить, что для тела такого размера и характерной скорости (например, «ядра») $F_d = k_2v^2$. Поскольку $v_x = v\cos\theta$, а $v_y =$

$= v \sin\theta$, уравнения (3.15) можно переписать в виде

$$\frac{dv_x}{dt} = -A v v_x , \tag{3.16а}$$

$$\frac{dv_y}{dt} = -g - A v v_y , \tag{3.16б}$$

где $A = k_2/m$. Заметим, что уравнения (3.16а) и (3.16б), описывающие изменения v_x и v_y, содержат модуль скорости: $v^2 = v_x^2 + v_y^2$. Следовательно, невозможно найти вертикальную составляющую движения падающего тела, не зная горизонтальную.

ЗАДАЧА 3.5. Траектория ядра

Модифицируйте программу **fall** так, чтобы вычислялась двумерная траектория шара с учетом сопротивления воздуха и можно было построить график функции y в зависимости от x. Например, подпрограмму **Euler** можно было бы написать следующим образом:

```
SUB Euler(x, y, vx, vy, t, A, g, dt, ncalc)          ! метод Эйлера-Кромера
  FOR icalc = 1 to ncalc
    LET v2 = vx*vx + vy*vy
    LET v = sqr(v2)
    LET ax = -A*v*vx
    LET ay = -g - A*v*vy
    LET vx = vx + ax*dt
    LET vy = vy + ay*dt
    LET x = x + vx*dt
    LET y = y + vy*dt
  NEXT icalc
  LET t = t + dt*ncalc
END SUB
```

а. В качестве проверки своей программы пренебрегите сначала сопротивлением воздуха, чтобы можно было сравнить получаемые результаты с известными. Например, представьте, что шар бросают с поверхности Земли под углом θ_0 к горизонту с начальной скоростью $v_0 = 15$ м/с. Проварьируйте величину угла θ_0 и покажите, что максимальное расстояние, на которое улетит шар, достигается при $\theta_0 = \theta_m = 45^\circ$. Чему равно максимальное расстояние R_m при броске

под таким углом? Сравните свое численное значение с точным ответом $R_m = v_0^2/g$.

б. Теперь предположим, что шар («ядро») бросают («толкают») с высоты h под углом θ_0 к горизонту с такой же начальной скоростью, как и в п. «а». Если пренебречь сопротивлением воздуха, какой, по вашему мнению, будет величина угла θ_m — больше или меньше 45°? Несмотря на то что эта задача решается аналитически, вы можете определить численное значение θ_m, не меняя своей программы. Чему равен угол θ_m для $h = 2h$? На сколько процентов изменяется дальность R, если угол θ отличается от θ на 2%?

в. Теперь рассмотрим влияние сопротивления воздуха на дальность полета и величину оптимального угла в задаче о толкании ядра. Для типичного ядра (масса ≈ 7кг, площадь сечения $\approx 0.01\text{м}^2$) параметр k_2, входящий в определение (3.8б), приблизительно равен 0.01. В каких единицах измеряется k_2? Сначала для удобства предположите, что влияние сопротивления воздуха очень велико, поскольку в этом случае можно представить качественную картину, отвлекаясь от конкретных числовых значений. Вычислите значение оптимального угла для случая $h = 2$ м, $v_0 = 30$ м/с и $A = k_2/m = 0.1$ и сравните его со значением, найденным в п. «б». Будет ли величина R более или менее чувствительна по сравнению со случаем «б» к изменениям угла θ_0 относительно угла θ_m? Определите оптимальный угол толчка и соответствующую дальность полета ядра для более реалистического значения величины $A = 0.001$. Задача о максимальной дальности полета ядра детально рассмотрена Лихтенбергом и Уиллсом (см. список литературы).

ЗАДАЧА 3.6. Связанное движение

Рассмотрим движение двух одинаковых тел, начинающих двигаться с высоты h. Одно из них начинает падать вертикально из состояния покоя, а другое бросают горизонтально со скоростью v_0. Какое из тел первым упадет на земную поверхность?

а. Обоснуйте физически свой ответ, считая, что сопротивлением воздуха можно пренебречь.

б. Предположите, что пренебречь сопротивлением воздуха нельзя и тормозящая сила пропорциональна v^2. Физически обоснуйте свой ответ. Затем проведите численное моделирование, используя, например, значения $A = k_2/m = 0.1$, $h = 10$ м с $v_0 = 0$ и $v_0 = 30$ м/с. Согласуются ли качественно ваши результаты с предварительным ответом? Если нет, то причиной расхождения может быть ошибка в вашей программе. Или же это расхождение, возможно, объясняется тем, что вы не чувствуете влияния связи вертикальной и горизонтальной составляющих движения.

в. Предположим, что тормозящая сила пропорциональна v, а не v^2. Будет ли ваш предварительный ответ таким же, как и в п. «б»? Проведите численные расчеты и проверьте свою интуицию.

3.6. ДРУГИЕ ПРИЛОЖЕНИЯ

Идеи и методы, которые мы обсудили, играют важную роль в физике облаков. Например, чтобы понять поведение падающих водяных капель, необходимо учитывать сопротивление воздуха наряду с ростом капель за счет конденсации и другими физическими механизмами. Ввиду разнообразия и сложности этих механизмов численное моделирование играет существенную роль в изучении таких процессов.

Другой областью приложений является изучение траекторий ядер различной формы, движущихся в воздухе. Особый интерес для спортивных болельщиков представляют траектории мячей, летящих с вращением, и влияние сопротивления воздуха на дальность полета и скорость шарика в настольном теннисе.

ЛИТЕРАТУРА

William R. Bennet, Scientific and Engineering Problem-Solving with the Computer, Prentice-Hall, 1976. Одна из первых, но до сих пор лучших книг, включающая численное решение задач. Рассматривается большое количество одномерных и двумерных задач о падении тел.

Byron L. Coulter, Carl G. Adler, Can a body falling through the air?, Am. J. Phys. **47**, 841 (1979). Авторы обсуждают предельные условия, при которых тормозящая сила линейна или квадратична по скорости.

R. M. Eisberg, Applied Mathematical Physics with Programmable Pocket Calculators, McGraw-Hill, 1976. В гл. 3 этой книги обсуждаются задачи о падении тел в том же духе, что у нас.

Richard P. Feynman, Robert B. Leighton, Matthew Sands, The Feynman Lectures on Physics, Vol. 1, Addison-Wesley, 1963. [Имеется перевод: *Фейнман Р., Лейтон Р., Сэндс М.*, Фейнмановские лекции по физике. — М.: Мир, 1966.] В гл. 9 Фейнман, в присущей только ему манере, рассматривает численное решение уравнений Ньютона.

A. P. French, Newtonian Mechanics, W. W. Norton & Company, 1971. В гл. 7 блестяще рассматривается вопрос о сопротивлении воздуха и подробно исследуется движение при наличии тормозящей силы.

Margaret Greenwood, Charls Hanna, John Milton, Air Resistence Acting on a Sphere: Numerical Analysis, Strobe Photographs and Videotapes, Phys. Teacher **24**, 153 (1986). Приводятся большое количество экспериментальных данных и теоретическое исследование движения шарика от настольного тенниса и из пенопласта. Табл. 3.1 приведена с разрешения авторов.

K. S. Krane, The falling raindrop: Variation on a theme of Newton, Am. J. Phys. **49**, 113 (1981). Авторы рассматривают задачу об аккреции массы каплей, движущейся сквозь облако.

D. B. Lichtenberg, J. G. Wills, Maximizing the range of the shot put, Am. J. Phys. **46**, 546 (1978). Задача 3.5 частично базируется на вопросах, которые обсуждаются в этой статье.

Rabindra Mehta, Aerodynamics of Sports Balls, in: Ann. Rev. Fluid Mech. **17**, 151 (1985).

R. R. Rogers, A Short Course in Cloud Physics, Pergamon Press, 1976.

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

Матвеев А. Н., Механика и теория относительности. — М.: Высшая школа, 1986. В § 36 рассматриваются вопросы падения тел в поле тяжести вблизи поверхности Земли и обсуждается влияние силы сопротивления воздуха, пропорциональной v^2. Изложение ведется на уровне, принятом в этой книге.

Аппель П., Теоретическая механика. — М.: ГИФМЛ, 1960. В гл. X подробно исследуются двумерные траектории тел и различные зависимости силы сопротивления воздуха от скорости. Изложены строгие результаты. Книга рассчитана на более подготовленного читателя.

ЗАДАЧА КЕПЛЕРА

4

Мы применяем законы движения Ньютона к движению планет и выделяем некоторые на первый взгляд неожиданные следствия из законов Ньютона.

4.1. ВВЕДЕНИЕ

Движение планет имеет особое значение, поскольку в прошлом оно сыграло важную роль в формировании механистического взгляда на Вселенную. Немногие теории оказали столь же огромное влияние на западную цивилизацию, как ньютоновы законы движения и всемирного тяготения, связывающие в единое целое движение звезд и земных объектов.

Большую часть наших знаний о движении планет объединили в себе законы Кеплера, которые можно сформулировать следующим образом:

1. Всякая планета движется по эллиптической орбите, в одном из фокусов которой находится Солнце.
2. Скорость планеты возрастает по мере удаления от Солнца таким образом, что прямая, соединяющая Солнце и планету, в равные промежутки времени заметает одинаковую площадь.
3. Для всех планет, вращающихся вокруг Солнца, отношение T^2/a^3 одинаково (T — период обращения планеты вокруг Солнца, a — большая полуось эллипса).

Кеплер вывел свои законы на основании тщательного анализа данных наблюдений, которые на протяжении многих лет собирал Тихо Браге.

Подчеркнем, что первый и третий законы Кеплера касаются *формы* орбиты, а не зависимости скорости и координаты планеты от времени. Поскольку эти временные зависимости невозможно получить в элементарных функциях, мы вынуждены рассматривать численное решение уравнений движения планет и их спутников по орбите. Кроме того, мы обсудим влияние возмущений на характер орбиты и рассмотрим некоторые задачи, которые бросают вызов нашей интуиции в правильном понимании законов движения Ньютона.

4.2. УРАВНЕНИЯ ДВИЖЕНИЯ ПЛАНЕТ

Движение Солнца и Земли является примером *задачи двух тел*. Эту задачу можно свести к задаче одного тела двумя методами. В основе самого простого метода лежит тот факт, что масса Солнца во много раз больше массы Земли. Следовательно, с хорошей точностью можно считать Солнце неподвижным и связать с ним начало системы координат. Если вы знакомы с понятием *приведенной массы*, то знаете, что суще-

ствует и более общий метод. А именно, движение двух тел с массами m и M, полная потенциальная энергия которых зависит только от расстояния между ними, можно свести к эквивалентной задаче о движении одного тела приведенной массы μ, определямой формулой

$$\mu = \frac{Mm}{m + M}. \tag{4.1}$$

Поскольку масса Земли $m = 5.99 \times 10^{24}$ кг, а масса Солнца $M = 1.99 \times \times 10^{30}$ кг, то понятно, что для большинства практических целей приведенная масса Солнца и Земли равна массе Земли. Поэтому ниже мы рассмотрим только задачу об одной материальной точке массой m, движущейся вокруг неподвижного силового центра, который мы примем за начало системы координат.

Закон всемирного тяготения Ньютона утверждает, что частица массой M притягивает другую частицу массой m с силой

$$\mathbf{F} = -\frac{GMm}{r^3}\mathbf{r}, \tag{4.2}$$

где вектор $\mathbf{r}$ направлен от тела с массой M к телу с массой m, а G — постоянная тяготения, которая, как экспериментально установлено, равна

$$G = 6.67 \cdot 10^{-11}\ \text{м}^3/\text{кг}\cdot\text{с}^2. \tag{4.3}$$

Отрицательный знак в формуле (4.2) означает, что гравитационная сила является силой притяжения, т.е. стремится уменьшить расстояние r между телами.

Закон (4.2) относится только к телам пренебрежимо малых пространственных размеров. Ньютон не публиковал свой закон всемирного тяготения 20 лет, хотя он изобрел интегральное исчисление и показал, что закон (4.2) применим также к любой однородной сфере или массовой сферической оболочке, если расстояние r измерять от центра каждой массы.

У силы тяготения имеются два свойства общего характера: ее величина зависит только от расстояния между телами, а направление совпадает с линией их соединяющей. Такие силы называются *центральными*. Из предположения о центральности силы следует, что орбита Земли лежит в плоскости $(x\text{–}y)$, а *угловой момент* $\mathbf{L}$ сохраняется и направлен по третьей оси (z). Запишем L_z в виде

$$L_z = (\mathbf{r} \cdot m\mathbf{v})_z = m(xv_y - yv_x), \tag{4.4}$$

где использовано определение векторного произведения $\mathbf{L} = \mathbf{r} \times \mathbf{p}$, а $\mathbf{p} = m\mathbf{v}$. Кроме того, движение ограничивается условием сохранения полной энергии E, равной

$$E = \frac{1}{2}mv^2 - \frac{GmM}{r}. \tag{4.5}$$

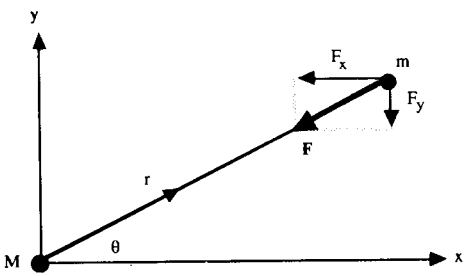

Рис. 4.1. Тело массой m движется под действием центральной силы F. Отметим, что выражения $\cos\theta = x/r$ и $\sin\theta = y/r$ позволяют записать уравнения движения в компонентах, что удобно для численного моделирования.

Если связать систему координат с телом массой M, то уравнение движения принимает вид

$$m\frac{d^2\mathbf{r}}{dt^2} = -\frac{mGM}{r^3}\mathbf{r}. \tag{4.6}$$

Для целей численного моделирования удобно записать силу в декартовых координатах (рис. 4.1):

$$F_x = -\frac{GMm}{r^2}\cos\theta = -\frac{mGM}{r^3}x, \tag{4.7а}$$

$$F_y = -\frac{GMm}{r^2}\sin\theta = -\frac{mGM}{r^3}y. \tag{4.7б}$$

В результате уравнения движения в декартовых координатах принимают вид

$$\frac{d^2x}{dt^2} = -\frac{GM}{r^3}x, \tag{4.8а}$$

$$\frac{d^2x}{dt^2} = -\frac{GM}{r^3}y, \tag{4.8б}$$

где $r^2 = x^2 + y^2$. Уравнения (4.8а) и (4.8б) — пример «системы дифференциальных уравнений», потому что каждое уравнение содержит как x, так и y.

1.3. ДВИЖЕНИЕ ПО ОКРУЖНОСТИ

Поскольку большинство орбит мало отличается от круговых, полезно получить условия движения тел по круговой орбите. Величина ускорения **a** связана с радиусом круговой орбиты r и скоростью тела v соотношением

$$a = \frac{v^2}{r}. \tag{4.9}$$

Ускорение всегда направлено к центру и обусловлено гравитационной силой. Следовательно, имеем

$$\frac{mv^2}{r} = \frac{mMG}{r^2} \tag{4.10}$$

или

$$v = \left(\frac{MG}{r}\right)^{1/2}. \tag{4.11}$$

Выражение (4.11), связывающее радиус и скорость, и есть общее условие любой круговой орбиты.

Можно также найти зависимость периода T от радиуса круговой орбиты. Используя соотношение

$$T = \frac{2\pi r}{v} \tag{4.12}$$

вместе с формулой (4.11), получим

$$T^2 = \frac{4\pi^2 r^3}{GM}. \tag{4.13}$$

Формула (4.13) представляет собой частный случай третьего закона Кеплера, поскольку радиус r соответствует большой полуоси эллипса.

4.4. ЭЛЛИПТИЧЕСКИЕ ОРБИТЫ

Поскольку известно, что наиболее общим видом орбиты является эллипс, подводя итог нашему обсуждению, опишем свойства эллиптической орбиты. Простое геометрическое определение параметров эллипса приведено на рис. 4.2. Оба *фокуса* эллипса, F_1 и F_2, обладают тем свойством, что для любой точки P, лежащей на этой кривой, сумма расстояний от фокусов $F_1P + F_2P$ постоянна. В общем случае у эллипса имеются две неравные взаимно перпендикулярные оси. Более длинная ось называется большой осью; половина этой оси — *большая полуось* a. Короткая ось называется малой осью эллипса; *малая полуось* b в два раза короче. В астрономии принято описывать эллиптическую орбиту величиной a и *эксцентриситетом* e, который равен отношению расстояния между фокусами к длине большей оси. Поскольку $F_1P + F_2P = 2a$, то легко показать (рассмотрев точку P с координатами $x = 0$, $y = b$), что

$$e = \sqrt{1 - \frac{b^2}{a^2}}\ , \tag{4.14}$$

причем $0 < e < 1$. В частном случае $b = a$ эллипс превращается в ок-

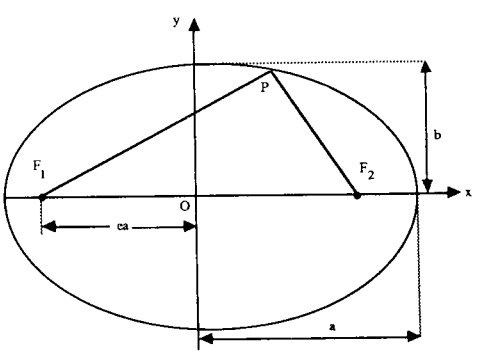

Рис. 4.2. Определение эллипса с помощью большой и малой полуосей a и b. Эксцентриситет e определен на рисунке. Начало декартовой системы координат O совпадает с центром эллипса.

ружность и $e = 0$. Величина эксцентриситета для орбиты Земли равна 0.0167.

4.5. АСТРОНОМИЧЕСКИЕ ЕДИНИЦЫ

Поскольку работать на компьютере с очень малыми или очень большими числами (например, G и M) по меньшей мере неудобно, желательно выбрать такую систему единиц, в которой величина произведения GM была бы порядка единицы. Для описания движения Земли принято в качестве единицы длины выбирать большую полуось земной орбиты. Эта единица длины называется *астрономической единицей* (а.е.), она равна

$$1 \text{ а.е.} = 1.496 \cdot 10^{11} \text{ м}. \tag{4.15}$$

В качестве единицы времени принимается один год $= 3.15 \cdot 10^{17}$ с. В этих единицах $T = 1$ год, $a = 1$ а.е., и можно записать

$$GM = \frac{4\pi^2 a}{T^2} = 4\pi^2 \,(\text{а.е.})^3/\text{год}^2. \tag{4.16}$$

4.6. ЗАМЕЧАНИЯ ПО ПРОГРАММИРОВАНИЮ

Поскольку мы убедимся в том, что для моделирования задачи Кеплера нет необходимости использовать более сложный алгоритм, структура программы решения системы уравнений движения (4.8) останется такой же, как в гл. 3. Главное изменение будет заключаться в применении массива для хранения двумерных координат и скоростей тела, движущегося по орбите. Описание размерности массива и передача массивов в подпрограмму иллюстрируется в программе **array**.

```
PROGRAM array                    ! иллюстрация использования массивов
DIM x(10),r(3,3)                 ! массивы описываются в инструкции DIM
CALL add(x,r)
FOR i = 1 to 10
   PRINT x(i);
NEXT i
PRINT
FOR i = 1 to 3
   FOR j = 1 to 3
      PRINT r(i,j);
   NEXT j
NEXT i
END

SUB add(x(),r(,))
  DIM y(-5 to 5)
  ! массивы можно описывать в головной программе или в подпрограмме
  FOR i = 1 to 10
     LET x(i) = i
  NEXT i
  FOR i = -5 to 5
     LET y(i) = i
     PRINT y(i)
  NEXT i
  FOR i = 1 to 3
     FOR j = 1 to 3
        LET r(i,j) = n
        LET n = n + 1
     NEXT j
  NEXT i
END SUB
```

В этой программе использованы следующие характерные особенности языка True BASIC:

1. Массивы описываются в инструкции **DIM**, при этом количество элементов массива указывается в скобках. Переменная x в головной программе и переменная y в подпрограмме **add** — примеры одномерных массивов; переменная r — пример двумерного массива.

2. Можно указывать нижний и верхний пределы каждого индекса массива; по умолчанию нижний предел равняется 1.
3. Массивы, как и другие переменные, можно передавать в подпрограммы или подпрограммы-функции. Скобки и запятые опускаются, если массив используется в качестве *фактического параметра* в инструкции **CALL**, например

```
CALL add(x,r)
```

Если массив является *формальным параметром*, то в этом случае используются пустые скобки и запятые, которые указывают размерность массива, например

```
SUB add(x(),r(,))
```

4. Заметим, что на самом деле весь массив не передается. Передается адрес первого элемента массива, а следовательно, передача массивов в подпрограмму никак не сказывается на объеме требуемой памяти и быстродействии.

Поскольку наблюдение движения тела по орбите весьма поучительно, то желательно написать подпрограмму, которая рисует его траекторию. Однако почти не бывает мониторов с квадратным экраном и одинаковым количеством пикселов по вертикали и по горизонтали. Поскольку желательно, чтобы круговая орбита и на экране выглядела как окружность, мы должны ввести поправку на характеристическое отношение (соотношение ширины и высоты изображения) своего экрана. Приведенная ниже программа вычерчивает окружность. Если кривая не будет выглядеть как окружность, внесите соответствующие исправления в инструкцию **SET window**.

```
PROGRAM circle                    ! проверка характеристического отношения
LET r = 1                         ! радиус окружности
! aspect_ratio равно отношению ширины экрана к его высоте
! aspect_ratio = 1.5              ! для компьютера Macintosh
INPUT prompt "характеристическое отношение = ":aspect_ratio
LET x = aspect_ratio*r
SET window -x,x,-r,r
BOX CIRCLE -r,r,-r,r
END
```

4.7. ЧИСЛЕННОЕ МОДЕЛИРОВАНИЕ ОРБИТЫ

В гл. 3 для моделирования падения тел поочередно использовались алгоритмы Эйлера и Эйлера—Кромера. [Результаты получились бы у вас лучше при использовании для вычисления x_{n+1} средней скорости $\frac{1}{2}(v_n + v_{n+1})$.] Как мы убедимся в дальнейшем, для того чтобы моделировать периодическое движение по эллиптической орбите с приемлемым шагом по времени, из рассмотренных нами алгоритмов только алгоритм Эйлера—Кромера дает устойчивые орбиты на протяжении многих периодов.

В программе **planet** используются три одномерных массива, содержащих по два элемента, каждый представляет соответственно координату, скорость и ускорение тела. Использование массива позволяет записать уравнение движения (4.8) в симметричной форме

```
accel(i) = -GM*pos(i)/(r*r*r)
vel(i) = vel(i)+accel(i)*vel(i)
pos(i) = pos(i)+vel(i)*dt
```

где *pos*(1) и *pos*(2) соответствуют координатам x и y. Отметим, что для получения численных значений координаты и скорости и для построения графика орбиты мы использовали отдельные подпрограммы.

```
PROGRAM planet                    ! движение планеты
! для нахождения новой координаты используется новая скорость
DIM pos(2), vel(2)                ! описание массивов
CALL initial(pos, vel, GM, dt, nplot, ncalc)
CALL output(pos)                  ! рисуется положение "земли"
FOR iplot = 1 to nplot
   CALL Euler(pos, vel, GM, dt, ncalc)
   CALL output(pos)
NEXT iplot
END
```

```
SUB initial(pos(),vel(),GM,dt,nplot,ncalc)
  LET GM = 4.0*pi*pi                    ! переход к астрономическим единицам
  INPUT prompt "шаг по времени = ":dt
  INPUT prompt "полное время = ":tmax          ! (годы)
  INPUT prompt "интервал вывода точек орбиты = ":plot_period ! (годы)
  LET ncalc = plot_period/dt      ! число шагов между точками орбиты
  LET nplot = tmax/plot_period
  INPUT prompt "начальная координата x = ": pos(1)
  LET pos(2) = 0                        ! начальная координата y
  LET vel(1) = 0                        ! начальная x-компонента скорости
  INPUT prompt "начальная y-компонента скорости = ":vel(2)
  LET r = 2*pos(1)                      ! допустимый максимум большой полуоси
  ! не квадратный экран
  ! характеристическое отношение (aspect_ratio) = ширина/высота (экрана)
  LET aspect_ratio = 1.5          ! значение для компьютера Macintosh
  LET x = aspect_ratio*r
  SET window -x,x,-r,r
  LET radius = 0.1                              ! радиус "солнца"
  BOX CIRCLE -radius,radius,-radius,radius ! "солнце" в начале координат
  FLOOD 0,0                                     ! "солнце" рисуется цветом знаков
END SUB

SUB Euler(pos(),vel(),GM,dt,ncalc)
  DIM accel(2)
  FOR icalc = 1 to ncalc
     LET r = sqr(pos(1)*pos(1)+pos(2)*pos(2))
     FOR i = 1 to 2
        LET accel(i) = -GM*pos(i)/(r*r*r)
        LET vel(i) = vel(i)+accel(i)*dt
        LET pos(i) = pos(i)+vel(i)*dt
     NEXT i
  NEXT icalc
END SUB

SUB output(pos())                       ! рисование орбиты
  PLOT POINTS: pos(1),pos(2)
END SUB
```

ЗАДАЧА 4.1. Проверка программы planet для круговых орбит

а. Проверьте правильность программы **planet**, рассматривая частный случай круговой орбиты. В качестве примера выберите (в астрономических единицах) $x_0 = 1$, $y_0 = 0$ и $v_x(t = 0) = 0$. С помощью соотношения (4.11) найдите численное значение $v_y(t = 0)$, которое дает круговую орбиту. Выберите значение шага Δt таким образом, чтобы полная энергия E сохранялась с хорошей точностью. (Заметим, что достаточно вычислять отношение E/m.) Достаточно ли мало значение Δt, выбранное вами, чтобы получаемая траектория воспроизводилась в течение многих периодов?

б. Пропустите программу **planet** с различными значениями начальных условий, x_0 и $v_y(t = 0)$, соответствующими условиям круговой орбиты. Задайте $y_0 = 0$ и $v_x(t = 0) = 0$. Для каждой орбиты определите радиус и период и проверьте правильность третьего закона Кеплера. Придумайте простое условие, которое позволит найти численное значение периода, и реализуйте его в отдельной подпрограмме.

в. Покажите, что метод Эйлера неустойчив для тех же значений шага, что и в п.п. «а» и «б». Достаточно ли просто уменьшить величину шага Δt или сам метод Эйлера неустойчив для данной динамической системы? Используйте для вычисления x_{n+1} среднюю скорость $\frac{1}{2}(v_n + v_{n+1})$. Будут ли эти результаты лучше всех остальных?

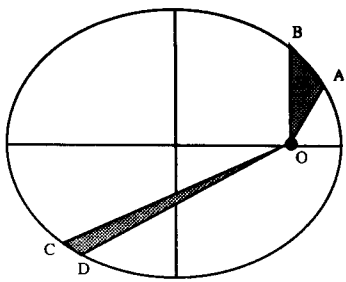

Рис. 4.3. Иллюстрация второго закона Кеплера равных площадей.

ЗАДАЧА 4.2. Проверка третьего закона Кеплера для эллиптических орбит

а. Задайте $y_0 = 0$ и $v_x(t = 0)$. Методом проб и ошибок подберите несколько вариантов значений x_0 и $v_y(t = 0)$, которые приводят к подходящим орбитам. Для каждой орбиты определите полную энергию, угловой момент, большую и малую полуоси, эксцентриситет и период обращения. Какой смысл имеет знак полной энергии? Если у вас имеется программа, непосредственно вычисляющая большую полуось, эксцентриситет и период обращения, оформите эту часть программы в виде отдельной подпрограммы. Воспользуйтесь своими данными о периоде и большой полуоси и проверьте третий закон Кеплера. В приложении 4А обсуждается использование логарифмических координат для построения степенных зависимостей.

б. Вы, вероятно, уже заметили, что алгоритм Эйлера—Кромера с постоянным шагом по времени перестает работать, если «планета» находится слишком близко к солнцу. Как вы можете наглядно подтвердить неработоспособность метода? Почему этот метод не работает? Предложите простую модификацию программы, которая может улучшить численные результаты.

ЗАДАЧА 4.3. Проверка второго закона Кеплера

а. Выберите такие начальные условия, чтобы получилась подходящая эллиптическая орбита. Поскольку расстояние между соседними «точками» орбиты является мерой скорости планеты, сначала определите качественно, как изменяется скорость планеты. В каких точках скорость максимальна (минимальна)?

б. Выберите точки орбиты A, B, C и D, как показано на рис. 4.3. Эти точки можно выбрать на орбите произвольным образом, но лучше, чтобы A и B находились вблизи одного конца большой полуоси, а C и D вблизи другого. Измерьте площади заштрихованных фигур OAB и OCD. (Один из способов вычисления площадей заключается в подкладывании под рисунок миллиметровой бумаги и подсчете числа клеточек, находящихся внутри каждой области.) Измерьте времена t_{AB} и t_{CD} движения планеты из точки A в B и из C в D и вычислите отношения OAB/t_{AB} и OCD/t_{CD}. Сравните эти результаты с теми, которые получаются из второго закона Кеплера.

ЗАДАЧА 4.4. Силы, которые не пропорциональны обратному квадрату

а. Рассмотрите динамические эффекты, обусловленные малым отклонением от закона обратных квадратов для силы притяжения, например $Km/r^{2+\delta}$, где $\delta = 0.05$. Как и прежде, для удобства положите постоянную K равной $4\pi^2$. Рассмотрите начальные условия $x_0 = 1$, $v_y(t = 0) = 5$ [как обычно, с $y_0 = 0$ и $v_x(t = 0) = 0$]. Вы обнаружите, что орбита планеты не является периодической. Убедитесь в том, что этот результат не зависит от выбора величины шага Δt. Будет ли планета двигаться по раскручивающейся или закручивающейся спирали вокруг солнца? Один из способов описания траектории — описание ее как эллиптической орбиты, которая медленно вращается или *прецессирует* в том же направлении, в котором движется планета. Удобно измерять прецессию углом между последовательными ориентациями большой полуоси эллипса. Этот угол равен скорости прецессии за оборот. Оцените величину этого угла для выбранных вами параметров орбиты и значения δ. Как влияет уменьшение большой полуоси? К какому результату приводит увеличение параметра δ?

б. Предположим, что сила притяжения обратно пропорциональна кубу расстояния, т.е. равна Km/r^3. В каких единицах будет измеряться K? Выберите численное значение $K = 4\pi^2$. Рассмотрите начальное условие $x_0 = 1$, $y_0 = 0$, $v_x(t = 0) = 0$ и аналитически вычислите значение $v_y(t = 0)$, необходимое для существования круговой орбиты. Насколько малым должен быть шаг Δt, чтобы алгоритм Эйлера — Кромера давал круговую орбиту в течение нескольких периодов? Как соотносится это значение Δt с тем, которое соответствует закону обратных квадратов?

в. Измените $v_y(t = 0)$ приблизительно на 2% по сравнению с условием для круговой орбиты, которое вы вычислили в п. «б». Какой будет новая орбита? Каков знак полной энергии? Будет ли орбита ограниченной? Замкнута ли она?

ЗАДАЧА 4.5. Влияние лобового сопротивления на орбиту спутника

Рассмотрим движение спутника по орбите вокруг Земли. В этой задаче удобно измерять расстояние в единицах радиуса Земли $R = 6.37 \cdot 10^6$ м. Время измеряется в часах. В таких земных едини-

цах (з.е.) численное значение постоянной тяготения G равно

$$Gm = 6.67 \cdot 10^{-11}\,\text{м}^3/\text{кг}\cdot\text{с}^2 \left(\frac{1\ \text{з.е.}}{6.37\cdot 10^6\,\text{м}}\right)^3 (3.6 \cdot 10^3\,\text{с}/\text{ч}^2)^2 =$$

$$= 3.34 \cdot 10^{-24}\,(\text{з.е.})^3/\text{кг}\cdot\text{ч}^2. \qquad (4.17)$$

Поскольку сила, действующая на спутник, пропорциональна Gm (m — масса Земли), то необходимо вычислить численное значение произведения Gm в земных единицах. Получим

$$Gm = 3.34 \cdot 10^{-24}\,(\text{з.е.})^3/\text{кг}\cdot\text{ч}^2 \cdot 5.99 \cdot 10^{24}\text{кг} =$$

$$= 20.0\,(\text{з.е.})^3/\text{ч}^2. \qquad (4.18)$$

Модифицируйте программу **planet**, чтобы учесть влияние силы сопротивления на движение спутника, вращающегося вокруг Земли. Выберите начальные условия так, чтобы в отсутствие сопротивления орбита была круговой, и позвольте спутнику сделать по меньшей мере один оборот, прежде чем «включить» силу сопротивления. Предположите, что сила сопротивления пропорциональна квадрату скорости спутника. Чтобы влияние сопротивления воздуха можно было увидеть за разумное время, предположите также, что величина тормозящей силы составляет приблизительно одну десятую от силы гравитационного взаимодействия. Опишите качественное изменение траектории спутника, обусловленное тормозящей силой. Как меняются со временем полная энергия и скорость спутника?

4.8. ВОЗМУЩЕНИЯ

Проверим, не подведет ли вас интуиция и глубоко ли вы чувствуете законы движения Ньютона. Для этого рассмотрим возмущения, налагаемые на стандартную задачу Кеплера. В каждом случае сначала ответьте на поставленные вопросы, а затем проводите численные расчеты.

ЗАДАЧА 4.6. Радиальные возмущения

а. Предположим, что спутник, движущийся по круговой орбите вокруг Земли, испытывает легкий «удар» или импульс силы в радиальном направлении (рис. 4.4,*а*). Как изменится в этом случае его орбита?

б. Как зависит это изменение от силы удара и его длительности?

в. После того как вы ответили на вопросы, поставленные в п.п. «а» и «б», проведите численные расчеты (см. программу key) и определите новую траекторию движения спутника. Выберите в качестве единиц измерения земные единицы (з.е.), чтобы численное значение произведения Gm задавалось выражением (4.18). Будет ли орбита *устойчивой*, например будет ли малый переданный импульс силы вызывать малое возмущение орбиты? Будет ли оставаться орбита периодической неопределенно долго, если никакие возмущения больше прилагаться не будут?

г. Установите, изменяются ли угловой момент и полная энергия спутника под действием радиального возмущения.

ЗАДАЧА 4.7. Тангенциальные возмущения

а. Предположим, что спутник, движущийся по круговой орбите вокруг Земли, испытывает легкий «удар» или импульс силы в касательном направлении (рис. 4.4,*б*). Как изменится в этом случае его орбита?

б. Как зависит это изменение от силы удара и его длительности?

в. После того, как вы ответили на вопросы, поставленные в п.п. «а» и «б», проведите численные расчеты и определите новую траекторию движения спутника. Будет ли орбита устойчивой?

г. Установите, изменяются ли угловой момент и полная энергия спутника под действием тангенциального возмущения.

д. Исследуйте устойчивость орбиты в случае силы, обратно пропорциональной кубу расстояния, под действием радиальных или тангенциальных возмущений.

Простой способ подействовать внешней силой в нужный момент времени — это воспользоваться инструкцией **key input**. Пример использования этой инструкции приводится в программе **key**. Пропустите эту про-

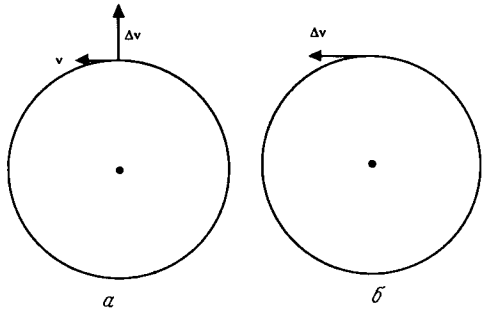

Рис. 4.4. Импульс силы приложен в радиальном (вертикальном) направлении (*а*). Импульс силы приложен в тангенциальном (горизонтальном) направлении (*б*).

грамму и определите, что получится в результате нажатия на клавиатуре клавиш, соответствующих буквам *k* или *K*.

```
PROGRAM key
FOR i = 1 to 10
  ! если клавиша нажата, key input равно "истина"
  IF key input then GET KEY kick
      IF kick = ord("k") or kick = ord("K") then
          PRINT kick
      ELSE
          PRINT "не нажата"
      END IF
  PAUSE 1
  LET kick = 0
NEXT i
END
```

Инструкция **key input** возвращает логическое значение «истина», если со времени последнего ввода с клавиатуры была нажата какая-либо клавиша. Инструкция **GET KEY** запоминает значение нажатой клавиши и используется в программе **key** для определения того, какая клавиша была нажата. Функция **ord** выдает значение, или ASCII-код, своего аргумента. Какие «значения» выдаст функция **ord** в случае строчной бук-

вы *k* и прописной буквы *K*?

Ниже приводится модифицированная версия подпрограммы **Euler**, которая позволяет учесть вертикальное или горизонтальное импульсное воздействие. Величина «удара» была выбрана для круговой орбиты с радиусом, равным 1, и шагом по времени 0.01.

```
SUB Euler(pos(), vel(), GM, dt, ncalc)          ! импульсное возмущение
  DIM accel(2), impulse(2)
  FOR icalc = 1 to ncalc
    LET impulse(1) = 0
    LET impulse(1) = 0
    LET kick = 0
    LET r = sqr(pos(1)*pos(1)+pos(2)*pos(2))
    IF key input then GET KEY kick
    IF (kick = ord("k")) or (kick = ord("K")) then
      LET magnitude = (Gm/r*r)          ! величина силы на единицу массы
      LET magnitude = 10*magnitude      ! величина "удара"
      LET impulse(2) = magnitude*dt ! вертикальный импульс на единицу массы
    END IF
    FOR i = 1 to 2
      LET accel = -GM*pos(i)/(r*r*r) + impulse(i)
      LET vel(i) = vel(i)+accel(i)*dt
      LET pos(i) = pos(i) + vel(i)*dt
    NEXT i
  NEXT icalc
END SUB
```

В этой версии подпрограммы **Euler** импульс силы приложен в вертикальном направлении. Следовательно, если мы хотим сообщить радиальный (тангенциальный) импульс силы, мы должны прикладывать его в тот момент, когда спутник находится в положении, показанном на рис. 4.4, *а* (рис. 4.4, *б*). В более общем случае импульс силы можно записать либо в виде

```
LET impulse(i) = magnitude*pos(i)/r          ! радиальный импульс
```

либо в виде

```
LET impulse(1) = -magnitude*pos(2)/r         ! тангенциальный импульс
LET impulse(2) = magnitude*pos(1)/r
```

В задачах 4.6 и 4.7 для моделирования влияния возмущений используйте любую версию подпрограммы **Euler**.

*ЗАДАЧА 4.8. Влияние «солнечного ветра»

Предположим, что на спутник помимо силы притяжения Земли действует в горизонтальном направлении слабая постоянная сила величиной W, обусловленная «солнечным ветром» (рис. 4.5). Уравнения

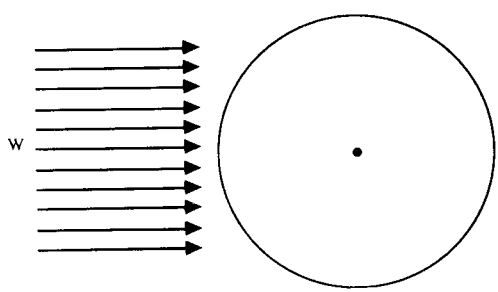

Рис. 4.5. Как изменится орбита под действием «солнечного ветра»?

движения можно записать в следующем виде:

$$\frac{d^2x}{dt^2} = -\frac{GMx}{r^3} + W, \tag{4.19а}$$

$$\frac{d^2x}{dt^2} = -\frac{GMy}{r^3}. \tag{4.19б}$$

Выберите начальные условия так, чтобы для $W = 0$ орбита была круговой. Затем положите **величину** W, равной приблизительно 3% от ускорения, обусловленного гравитационным полем, и численно промоделируйте орбиту. Как изменится орбита? (Подробное рассмотрение этой задачи можно посмотреть в статье Люерманна.)

4.9. ПРОСТРАНСТВО СКОРОСТЕЙ

Возможно, что в рассмотренных выше случаях интуиция вас подвела. Например, в задачах 4.6 и 4.7 вы, быть может, подумали, что орбита будет вытягиваться в направлении действия удара. Да, действительно, орбита вытягивается, но в направлении, *перпендикулярном* удару. Не переживайте, вы среди своих! Очень мало студентов хорошо понимают на качественном уровне законы движения Ньютона (см. статью Мак Клоски). Одна из качественных формулировок второго закона Ньютона звучит так:

Силы действуют на траектории частиц, изменяя скорость, а не координату.

Если не учитывать это обстоятельство, то можно столкнуться с физическими ситуациями, которые явно противоречат здравому смыслу.

Поскольку сила действует непосредственно на изменение скорости, имеет смысл рассматривать скорость и координату в одном базисе. Фактически в современном изложении классической механики и в квантовой механике обе переменные и рассматриваются таким образом.

Теперь «откроем» некоторые свойства орбит в пространстве «скоростей» тем же способом, как мы проделали это для орбит в пространстве «координат». Модифицируйте свою программу так, чтобы построить траекторию Земли в пространстве скоростей, т.е. так же, как строили график $\{x,y\}$, постройте и график $\{v_x,v_y\}$. Траектория в пространстве скоростей представляет собой ряд последовательных значений вектора скорости тела. Если в системе пространственных координат орбита является эллипсом, какую форму имеет орбита в пространстве скоростей? Мы рассматриваем такие вопросы, предполагая, что движение ограничено и происходит в поле силы, обратно пропорциональной квадрату расстояния. Подробное рассмотрение орбит в пространстве скоростей имеется в статье Абельсона и др.

ЗАДАЧА 4.9. Свойства орбит в пространстве скоростей

а. Убедитесь в том, что орбита в пространстве скоростей является окружностью (хотя в пространственных координатах она эллиптическая). Совпадает ли центр этой окружности с началом координат $\{v_x, v_y\} = \{0,0\}$ в пространстве скоростей? Рассмотрите как эл-

липтическую, так и круговую орбиты в системе пространственных координат. Неплохо выбрать те же самые начальные условия, что и в задачах 4.1 и 4.2.

б. Пусть **u** обозначает радиус-вектор точки, лежащей на окружности в пространстве скоростей, а **w** обозначает вектор, направленный из начала координат пространства скоростей в центр этой окружности (рис. 4.6). Тогда скорость частицы можно записать в виде

$$\mathbf{v} = \mathbf{u} + \mathbf{w}. \tag{4.20}$$

Вычислите величину u и убедитесь в том, что она определяется формулой

$$u = GmM/L, \tag{4.21}$$

где L — величина углового момента. Заметим, что величина L пропорциональна массе m, поэтому само значение массы m знать необязательно.

в. Убедитесь в том, что в каждый момент времени радиус-вектор планеты **r** перпендикулярен скорости **u**. Объясните этот факт.

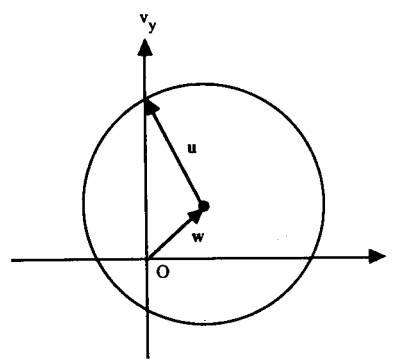

Рис. 4.6. Орбита материальной точки в пространстве скоростей. Вектор **w** направлен из начала координат пространства скоростей в центр круговой орбиты. Вектор **u** направлен из центра орбиты в точку с координатами (v_x, v_y).

ЗАДАЧА 4.10. Возмущения в пространстве скоростей

Как изменяется орбита в пространстве скоростей под действием импульсного удара в тангенциальном направлении? Тот же вопрос для случая, когда импульс действует в радиальном направлении? Как изменятся длина и направление вектора **w**? Исходя из наблюдаемых изменений орбиты в пространстве скоростей и вышеприведенных соображений, объясните наблюдаемое изменение орбиты в системе пространственных координат.

***ЗАДАЧА 4.11. Орбиты с учетом солнечного ветра**

Определите, как изменяется орбита в пространстве скоростей под действием солнечного ветра. Как меняются полный угловой момент и энергия? Приведите простое объяснение ранее обнаруженных изменений орбиты в системе пространственных координат.

4.10. СОЛНЕЧНАЯ СИСТЕМА В МИНИАТЮРЕ

До сих пор наше численное моделирование движения планет по орбитам ограничивалось задачей двух тел в поле центральных сил. Однако Солнечная система не является системой двух тел, поскольку между всеми планетами действуют гравитационные силы. Несмотря на то что силы взаимодействия между планетами малы по сравнению с гравитационной силой Солнца, они могут приводить к наблюдаемым эффектам. Например, существование планеты Нептун было предсказано на основании несовпадения экспериментально измеренной орбиты Урана и предсказанной орбиты, рассчитанной на основе известных сил.

Присутствие других планет означает, что полная сила, действующая на каждую планету, уже не является центральной. Более того, поскольку орбиты планет не лежат строго в одной плоскости, исследование Солнечной системы, если необходимы точные расчеты, должно проводится в трехмерной геометрии. Для простоты мы рассмотрим модель двумерной Солнечной системы, состоящей из двух планет, которые вращаются вокруг Солнца.

Уравнения движения двух планет с массами m_1 и m_2 можно записать

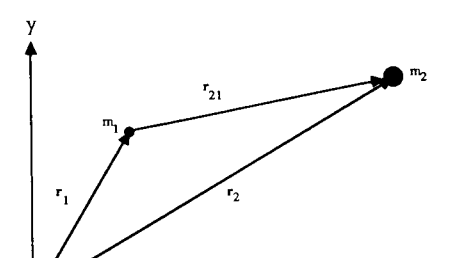

Рис. 4.7. Система координат, применяемая в задаче (4.22). Планеты с массами m_1 и m_2 обращаются вокруг «солнца» с массой M.

в векторной форме следующим образом (рис. 4.7):

$$m_1 \frac{d^2 \mathbf{r}_1}{dt^2} = -\frac{m_1 MG}{r_1^3}\mathbf{r}_1 + \frac{m_1 m_2 G}{r_{21}^3}\mathbf{r}_{21} \qquad (4.22а)$$

и

$$m_1 \frac{d^2 \mathbf{r}_2}{dt^2} = \frac{m_2 MG}{r_2^3}\mathbf{r}_2 - \frac{m_1 m_2 G}{r_{21}^3}\mathbf{r}_{21}, \qquad (4.22б)$$

где $\mathbf{r}_1$ и $\mathbf{r}_2$ — радиус-векторы, направленные от солнца к планетам 1 и 2, а $\mathbf{r}_{21} = \mathbf{r}_2 - \mathbf{r}_1$ — вектор, направленный от планеты 1 к планете 2. Несмотря на то что не существует никакого аналитического решения уравнений (4.22), численное решение можно получить простым обобщением предыдущего метода. В программе **planet2** координаты, скорости и ускорения этих двух планет содержатся в двумерных массивах, у которых первый индекс соответствует номеру планеты, а второй представляет компоненту x или y соответствующего вектора. Мы выбираем астрономические единицы, так что произведение GM задается выражением (4.16). Для иллюстративных целей выбираем численные значения m_1/M = = 0.001 и m_2/M = 0.01.

```
PROGRAM planet2 ! двумерная солнечная система с большой и малой планетами
DIM pos(2,2),vel(2,2)
CALL initial(pos,vel,GM,dt,ncalc)
CALL output(pos)
DO until key input
  CALL Euler(pos,vel,GM,dt,ncalc)
  CALL output(pos)
LOOP
END

SUB initial(pos(,),vel(,),GM,dt,ncalc)
  LET GM = 4.0*pi*pi                ! астрономические единицы
  LET dt = 0.001                    ! шаг по времени (годы)
  LET plot_period = 0.02            ! годы
  LET ncalc = plot_period/dt
  ! первая планета
  LET pos(1,1) = 1                  ! начальная координата x планеты 1
  LET pos(1,2) = 0                  ! начальная координата y планеты 1
  LET vel(1,1) = 0                  ! начальная x-компонента скорости
  LET vel(1,2) = sqr(GM/pos(1,1))   ! начальная y-компонента скорости
  ! вторая планета
  LET pos(2,1) = (4)^(1/3)          ! начальная координата x планеты 2
  LET pos(2,2) = 0                  ! начальная координата y планеты 2
  LET vel(2,1) = 0                  ! начальная x-компонента скорости
  LET vel(2,2) = sqr(GM/pos(2,1))   ! начальная y-компонента скорости
  LET r = 2*pos(2,1)
  LET aspect_ratio = 1.5
  LET x = aspect_ratio*r
  SET window -x,x,-r,r
  LET radius = 0.1
  BOX CIRCLE -radius,radius,-radius,radius  ! солнце в начале координат
  FLOOD 0,0
END SUB
```

```
SUB Euler(pos(,),vel(,),GM,dt,ncalc)
  DIM a(2,2),r(2)
  FOR icalc = 1 to ncalc
    ! вычисление расстояния dr между планетами 1 и 2
    LET dx = pos(2,1) - pos(1,1)
    LET dy = pos(2,2) - pos(1,2)
    LET dr = sqr(dx*dx + dy*dy)  ! расстояние между планетами 1 и 2
    LET accel = GM/(dr*dr*dr)
    LET a(1,1) = -0.01*accel*dx   ! ускорение планеты 1 из-за планеты 2
    LET a(1,2) = -0.01*accel*dy
    LET a(2,1) = -0.001*a(1,1)    ! ускорение планеты 2 из-за планеты 1
    LET a(2,2) = -0.001*a(1,2)
    FOR iplanet = 1 to 2              ! суммирование по планетам
      LET dist2 = pos(iplanet,1)*pos(iplanet,1)
      LET dist2 = dist2+pos(iplanet,2)*pos(iplanet,2)
      ! расстояние планеты от солнца
      LET r(iplanet) = sqr(dist2)
      FOR i = 1 to 2                  ! суммирование по компонентам
        LET r3 = r(iplanet)*r(iplanet)*r(iplanet)
        LET accel = -GM*pos(iplanet,i)/r3
        LET accel = accel+a(iplanet,i)
        LET vel(iplanet,i) = vel(iplanet,i) + accel*dt
        LET pos(iplanet,i) = pos(iplanet,i) + vel(iplanet,i)*dt
      NEXT i
    NEXT iplanet
  NEXT icalc
END SUB
SUB output(pos(,))                    ! рисование орбит
  PLOT POINTS: pos(1,1),pos(1,2)      ! первая планета
  PLOT POINTS: pos(2,1),pos(2,2)      ! вторая планета
END SUB
```

*ЗАДАЧА 4.12. Возмущения движения планет

а. Используйте программу **planet2** с теми начальными условиями, которые в ней заложены. Какими были бы орбиты и периоды обеих планет, если бы они не взаимодействовали между собой? В чем качественно проявляется их взаимодействие? Чем обусловлено то, что влияние взаимодействия на одну из планет меньше, чем на другую?

Опишите форму обеих орбит. Сохраняются ли полная энергия и угловой момент первой планеты? Сохраняются ли полная энергия и угловой момент обеих планет? Если вам интересно, то вы, возможно, захотите воспользоваться реальными астрономическими данными по Земле и Юпитеру и определить, оказывает ли какое-либо влияние Юпитер на орбиту Земли.

6. Другой интересной динамической системой является движение планеты в поле тяготения двух неподвижных звезд одинаковой массы. В этом случае никаких замкнутых орбит не существует, но все орбиты можно разделить на устойчивые и неустойчивые. Устойчивыми орбитами могут быть открытые петли, охватывающие обе звезды, орбиты в виде восьмерок или кеплероподобные орбиты вокруг только одной звезды. В случае неустойчивых орбит планета в конце концов упадет на одну из звезд.

ЛИТЕРАТУРА

Harold Abelson, Andrea diSessa, Lee Rudolph, Velosity Space and the Geometry of Planetary Orbits, Am. J. Phys., **43**, 579 (1975). См. также *Andrea diSessa*, Orbit: A mini-environment for exploring orbital mechanics, in: Computers in Education, eds. O. Lecarme, R. Lewis, 359, North-Holland, 1975. Представлено подробное без использования дифференциального исчисления чисто геометрическое рассмотрение происхождения замкнутых орбит в случае сил, обратно пропорциональных квадрату расстояния. Проще ли понять геометрические доказательства, чем алгебраические выкладки?

Ralph Baierlein, Newtonian Dynamics, McGraw-Hill, 1983. Средней сложности книга по механике. Особый интерес представляет обсуждение устойчивости круговых орбит и влияние сплющенности Солнца.

Alan Cromer, Stable solutions using the Euler approximation, Am. J. Phys., **49**, 455 (1981). Автор показал, что незначительное изменение обычного алгоритма Эйлера приводит к устойчивым решениям для периодических систем, в том числе для движения планет и гармонического осциллятора.

Alan Cromer, Computer-Simulated Physics Experiments, EduTech, North-eastern University, 1980. Руководство по лабораторным работам, ориентированное на использование компьютера Apple 11^+.

Robert M. Eisberg, Lawrence S. Lerner, Physics, Vol. 1, McGraw-

Hill, 1981. Книга для начинающих, в которой движение планет рассматривается с использованием численных методов.

A. P. French, Newtonian Mechanics, W.W. Norton & Company, 1971. Книга для начинающих, в которой более подробно, чем обычно, рассматривается движение планет.

Herbert Goldstein, Classical Mechanics, 2nd ed., Addisin–Wesley, 1980. [Имеется перевод: *Г. Голдстейн*, Классическая механика. — М.: Мир, 1974.] В гл. 3 дано прекрасное обсуждение задачи Кеплера и условий, при которых существует замкнутая орбита.

Arthur W. Luehrman, Orbits in the Solar Wind — a Mini-Research Problem, Am. J. Phys., **42**, 361 (1974). Автор подчеркивает необходимость студенческих задач, требующих индуктивной, а не дедуктивной аргументации.

Michael McCloskey, Intuitive Physics, Sci. Amer. **249**, 122 (April 1983). Обсуждение «неинтуитивной» сути законов Ньютона.

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

Матвеев А. Н., Механика и теория относительности. — М.: Высшая школа, 1986. Гл. 11 посвящена тяготению. В § 44 рассматриваются законы Кеплера, в § 46 — задача двух тел.

Аппель П., Теоретическая механика. — М.: ГИФМЛ, 1960. В гл. XI «Центральные силы. Эллиптическое движение планет» подробно исследуются вопросы, рассматриваемые авторами этой книги, в том числе силы обратно пропорциональные различным степеням r. Рассчитана на подготовленного читателя.

ПРИЛОЖЕНИЕ 4А. ГРАФИКИ В ЛОГАРИФМИЧЕСКОМ МАСШТАБЕ

В задаче 4.2 мы получили «экспериментальные данные» для периода и большой полуоси некоторых эллиптических орбит. Типичные данные для T и a приведены в табл. 4.1. Обычно нам надо анализировать данные, подобные приведенным в таблице, чтобы выяснить, удовлетворяют ли измеренные величины какой-то конкретной математической зависимости. Ниже мы проведем анализ данных из табл. 4.1 и обсудим, как такие зависимости можно было бы найти в некоторых простых случаях.

Предположим, нам необходимо проверить, удовлетворяют ли две переменные x и y некоторому заданному функциональному соотношению $y = f(x)$. Чтобы не усложнять исследование, пренебрежем возможными ошибками измерения величин x и y. (Аппроксимация данных при наличии ошибок рассматривается в приложении 11А.) Самой простой зависимостью двух переменных величин x и y является линейная, т.е. $y = mx + b$. Наличие этой зависимости можно увидеть, построив на простой миллиметровой бумаге y как функцию от x. Всегда стоит начинать с построения такого графика, если только переменные x или y не меняются на много порядков по величине.

ТАБЛИЦА 4.1. Данные для периода T и большой полуоси a из задачи 4.2. Использован шаг по времени $\Delta t = 0.01$. Оцененная погрешность T и a равна ± 0.02

T	a
0.50	0.62
0.63	0.73
2.18	1.18
3.44	2.28
8.95	4.31

Из табл. 4.1 видно, что T не является линейной функцией от a. В зависимости от характера задачи, возможно, имеются основания предполагать существование зависимости общего вида

$$y(x) = Ae^{rx} \tag{4.23}$$

или

$$y(x) = Ax^r, \tag{4.24}$$

где A и r — неизвестные параметры. Если принять зависимость вида (4.23), то можно прологарифмировать обе части формулы и тогда получаем $\ln y = \ln A + r\ln x$. Заметим, что если применима формула (4.23), то график зависимости $\ln y$ от x будет представлять собой прямую с угловым коэффициентом r, пересекающую ось ординат в точке $\ln A$. Обычно для построения такой зависимости используется полулогарифмическая бумага, у которой вертикальная ось размечена в логарифмическом масштабе, а горизонтальная ось — в обычном линейном. Можно действовать и по-другому: строить график $\ln y$ в зависимости от x на простой миллиметровке.

Если мы хотим попробовать описать данные формулой (4.24), можно прологарифмировать обе части и получить $\ln y = \ln A + \ln x$. В этом случае график y как функции от x, построенный в дважды логарифмическом масштабе, дает угловой коэффициент r, который нас и интересует. Конечно, можно строить также график $\ln y$ в зависимости от $\ln x$ и на обычной миллиметровке.

Проиллюстрируем простой анализ данных из табл. 4.1. На рис. 4.8 изображен график зависимости $\ln T$ от $\ln a$. Внешний вид графика указывает на то, что связь $\ln T$ и $\ln a$ вполне похожа на линейную. В гл. 11 будет обсуждаться метод «наименьших квадратов», позволяющий провести прямую через ряд экспериментальных точек. Однако, попрактиковавшись немного, вы сможете почти с тем же успехом проводить визуальный анализ данных. Угловой коэффициент, определенный из гра-

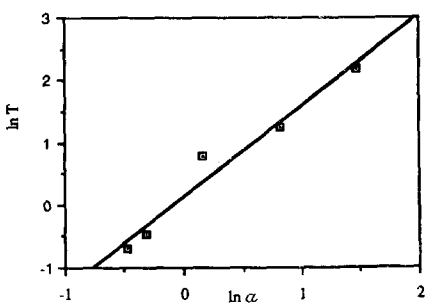

Рис. 4.8. График $\ln T$ в зависимости от $\ln a$, построенный по данным из табл. 4.1. Угловой коэффициент приближенно равен 1.46.

фика, приблизительно равен 1.46, что согласуется с теоретическим предсказанием, дающим значение 1.5.

Простой график зависимости $\ln T$ от $\ln a$, приведенный на рис.4.8, дает *полное* представление о характере связи между T и a. Однако теперь, когда мы уверены, что установили линейную зависимость, полезно посмотреть на график зависимости $\ln T/\ln a$ от $\ln a$ (рис. 4.9), которая гораздо более чувствительна к расхождениям с теорией и погрешностям.

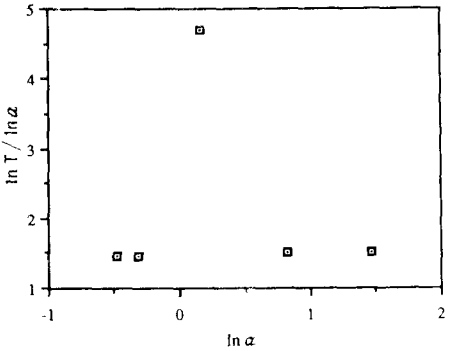

Рис. 4.9. График зависимости отношения $\ln T/\ln a$ от $\ln a$. Не кажется ли вам, что была допущена некоторая небрежность в выборе данных?

Разумеется, не все функциональные зависимости имеют вид простых степенных или показательных функций. Несомненно, что для проведения анализа данных требуется еще и некоторая теоретическая информация.

ЛИТЕРАТУРА

G. L. Squires, Practical Physics, 3rd ed., Cambridge University Press, 1985. Великолепная книга по планированию экспериментов и анализу данных.

КОЛЕБАНИЯ

5

В этой главе изучается поведение линейных и нелинейных колебательных систем, которые встречаются в механике и электронике.

5.1. ПРОСТОЙ ГАРМОНИЧЕСКИЙ ОСЦИЛЛЯТОР

Во многих физических системах движение носит регулярный периодический характер. Движение, которое повторяется через конечный интервал времени, например движение Земли вокруг Солнца, называют *периодическим*, или *гармоническим*. В том случае, когда объект движется по одной траектории между двумя предельными положениями, движение называют *колебанием*. Примерами колебаний, с которыми мы встречаемся в повседневной жизни, служат маятник в дедушкиных часах и звучащая гитарная струна. Менее очевидными примерами являются микроскопические явления, такие как колебания атомов в кристаллах. Большое количество немеханических примеров можно привести из области электромагнитных и атомных явлений.

Для ознакомления с основными понятиями, связанными с простыми колебаниями, рассмотрим тело массой m, прикрепленное к свободному концу пружины. Тело скользит по горизонтальной поверхности без трения (рис. 5.1). Будем описывать положение тела координатой x и примем точку $x = 0$ в качестве положения *равновесия*, т.е. положения,

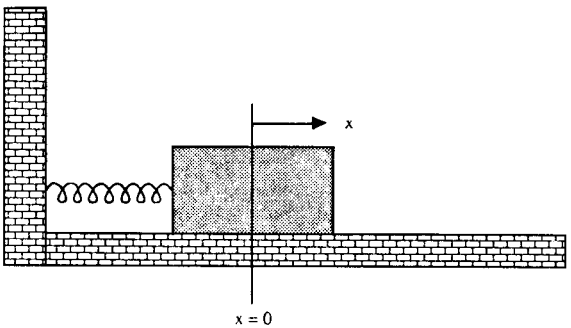

Рис. 5.1. Пример одномерного простого гармонического осциллятора. Кирпич скользит без трения по горизонтальной поверхности.

в котором пружина не напряжена. Если тело сместить из положения равновесия $x = 0$ и затем отпустить, то оно будет колебаться в горизонтальном направлении. Известно, что если пружина не слишком сильно растянута или сжата, то сила, действующая на тело с координатой x, является линейной относительно x:

$$F = -kx. \tag{5.1}$$

Силовая постоянная k является мерой жесткости пружины. Знак минус в (5.1) указывает на то, что сила стремится вернуть тело в положение равновесия. Уравнение движения этого тела можно записать в виде

$$\frac{d^2x}{dt^2} = -\omega_0^2 x, \tag{5.2}$$

где величина ω_0 определяется выражением

$$\omega_0^2 = \frac{k}{m}. \tag{5.3}$$

Уравнение (5.2) — пример *линейного* дифференциального уравнения, поскольку в него входят только первые степени переменной x и ее производных. Движение, описываемое уравнением (5.2), называется *простым гармоническим колебанием*, и его решение можно выразить аналитически через синусы и косинусы. Поскольку вид данного решения поможет нам ввести некоторые понятия, необходимые для изучения колебаний, приведем здесь решение этого уравнения. Одна из формул решения имеет вид

$$x(t) = A\cos(\omega_0 t + \delta), \tag{5.4}$$

где A и δ — постоянные, а аргумент косинуса выражается в радианах. Прямой подстановкой (5.4) в (5.2) убеждаемся в том, что выражение (5.4) является решением. Постоянные A и δ называются *амплитудой* и начальной *фазой* и могут быть определены из начальных условий для координаты x и скорости $v = dx/dt$.

Поскольку косинус является периодической функцией с периодом 2π, понятно, что функция $x(t)$ в выражении (5.4) также периодическая. Определим *период* T как наименьшее время, через которое движение повторяется, т.е.

$$x(t+T) = x(t). \tag{5.5}$$

Поскольку $\omega_0 T$ соответствует одному *периоду*, получим

$$T = \frac{2\pi}{\omega_0} = \frac{2\pi}{\sqrt{k/m}}. \tag{5.6}$$

Частота колебаний ν представляет собой число периодов в секунду и

определяется выражением $\nu = 1/T$. Заметим, что T зависит от отношения k/m и не зависит от A и δ. Следовательно, период простых гармонических колебаний не зависит от их амплитуды.

Хотя координата и скорость осциллятора непрерывно изменяются, полная энергия E остается постоянной и равна

$$E = \frac{1}{2}mv^2 + \frac{1}{2}kx^2. \tag{5.7}$$

Эти два члена в формуле (5.7) можно отождествить соответственно с кинетической и потенциальной энергией.

5.2. ЧИСЛЕННОЕ МОДЕЛИРОВАНИЕ ГАРМОНИЧЕСКОГО ОСЦИЛЛЯТОРА

Хорошо нам уже известный алгоритм Эйлера—Кромера особенно подходит для моделирования колебаний. Краткое изложение некоторых широко распространенных численных методов «высокого порядка» приводится в приложении 5А.

В программе **sho** с помощью алгоритма Эйлера—Кромера вычисляются зависимости координаты и скорости простого гармонического осциллятора от времени. Результаты вычисления координаты *pos* и скорости *vel* выводятся на экран до тех пор, пока работа программы не прерывается нажатием любой клавиши. Примеры использования функции **tab** и функций **PRINT using** приводятся соответственно в подпрограммах ***initial*** и ***output***.

```
PROGRAM sho                          ! простой гармонический осциллятор
CALL initial(pos, vel, w2, dt, ncalc)
DO until key input
   CALL output(pos, vel, t)
   CALL Euler(pos, vel, w2, dt, ncalc)
   LET t = t + ncalc*dt
LOOP
END
```

```
SUB initial(pos, vel, w2, dt, ncalc)
  INPUT prompt "начальная координата = ": pos  ! метры
  LET vel = 0                                  ! начальная скорость (м/с)
  INPUT prompt "отношение k/m =": w2   ! собственная (угловая) частота
  INPUT prompt "шаг по времени (сек) = ": dt
  LET print_period = 0.05
  LET ncalc = print_period/dt
  PRINT tab(7);"время";tab(17);"координата";tab(28);"скорость"
  PRINT                                        ! пропуск строки
  END SUB

  SUB Euler(pos, vel, w2, dt, ncalc)           ! алгоритм Эйлера-Кромера
  FOR icalc = 1 to ncalc
     LET accel = -w2*pos
     LET vel = vel + accel*dt
     LET pos = pos + vel*dt
  NEXT icalc
END SUB

SUB output(pos, vel, t)
  ! - перед числом печатается пробел или знак минус
  ! % незначащие нули в начале числа печатаются в виде одного '0'
  PRINT using "----%.****": t, pos, vel
END SUB
```

Сначала убедимся в том, что при надлежащем выборе шага Δt алгоритм Эйлера—Кромера можно использовать для описания движения простого гармонического осциллятора. Поскольку нам известно аналитическое решение для этого случая, то одно из условий выбора значения Δt заключается в том, что вычисленные решения должны соответствовать аналитическим результатам (5.4). Однако нам нужно какое-то более общее условие, которое не связано с существованием аналитического решения. В следующей задаче мы установим, что необходимым условием является сохранение полной энергии.

ЗАДАЧА 5.1. Сохранение энергии и выбор алгоритмов

а. Модифицируйте программу **sho** так, чтобы полная энергия, отнесенная к единице массы, E_n вычислялась на каждом временно́м шаге t_n. Вычислите и постройте график временно́й зависимости величины

$\Delta_n = (E_n - E_0)/E_0$ по крайней мере для одного периода (E_0 — начальная полная энергия). Остается ли разность Δ_n в течение всего периода малой величиной? Для простоты выберите для первых расчетов следующие параметры: $x_0 = 1$, $v_0 = 0$ и $\omega_0^2 = k/m = 9$. В каких единицах измеряется k и отношение k/m в системе СИ?

б. Воспользуйтесь методом Эйлера и вычислите Δ_n на одном периоде. Качественно объясните различия зависимостей Δ_n от времени, полученных при использовании методов Эйлера и Эйлера—Кромера. Какой метод предпочтительнее с точки зрения сохранения энергии?

в. Какой из методов дает при данном Δt лучшие результаты для координаты и скорости в смысле их близости аналитическому решению (5.4)? Соответствует *ли* требование сохранения энергии относительной точности вычисления координат? Для остальных задач этой главы примите метод Эйлера или Эйлера—Кромера.

г. Выберите подходящие численные значения шага Δt для случаев $\omega_0^2 = 9$ и $\omega_0^2 = 90$. Как эти значения Δt соотносятся с относительной величиной периода в этих двух случаях?

*д. Воспользуйтесь одним из алгоритмов «более высокого» порядка, обсуждаемых в приложении 5А для моделирования движения простого гармонического осциллятора. Является ли хоть один из этих алгоритмов более предпочтительным, чем алгоритм Эйлера—Кромера?

ЗАДАЧА 5.2. Анализ простого гармонического движения

а. Модифицируйте программу **sho** таким образом, чтобы строились зависимости координаты и скорости осциллятора от времени t. Качественно опишите поведение координаты и скорости осциллятора.

б. Для различных ω_0 вычислите соответствующие им значения периодов T. Как вы вычисляете период T? Предположите, что величина T пропорциональна $(k/m)^{\alpha}$, и оцените показатель степени α, построив график зависимости T от (k/m) в дважды логарифмическом масштабе (см. приложение 4А).

в. Вычислите значения амплитуды A и полной энергии E для начальных условий $x_0 = 4$, $v_0 = 0$ и $x_0 = 0$, $v_0 = 4$, выбрав в обоих случаях $\omega_0^2 = 4$. Какая величина определяет значение амплитуды A?

г. Постройте временны́е зависимости потенциальной и кинетической энергий за один период. Где кинетическая энергия достигает максимального значения?

д. Вычислите средние значения потенциальной и кинетической энергий за полный период. Существует ли какая-то связь между двумя этими величинами?

е. Вычислите функцию $x(t)$ для различных значений A и покажите, что форма этой функции не зависит от A, т.е. $x(t)/A$ является *универсальной* функцией времени для данного значения k/m. В каких единицах необходимо измерять время, чтобы величина $x(t)/A$ не зависела от A и k/m?

*ж. Нам известно, что движение одномерного осциллятора полностью определено, если заданы две функции $x(t)$ и $v(t)$. Значения этих функций можно рассматривать как координаты некоторой точки в двумерном пространстве, которое называется *фазовой плоскостью*. По мере увеличения времени точка $\{x,v\}$ описывает на фазовой плоскости некоторую траекторию. Модифицируйте свою программу так, чтобы строился график зависимости v от x. По вертикальной и горизонтальной осям откладывайте соответственно v и x. Положите $\omega_0^2 = 9$ и рассчитайте фазовые траектории для различных начальных условий $\{x_0 = 1, \quad v_0 = 0\}$, $\{x_0 = 0, \quad v_0 = 1\}$ и $\{x_0 = 4, \quad v_0 = 0\}$. Будут ли отличаться фазовые траектории для различных начальных условий? Какая физическая величина определяет отличие фазовых траекторий? Какой вид имеют фазовые траектории? Похожи ли друг на друга все фазовые траектории? Всегда ли типичная точка $\{x, v\}$ движется по или против часовой стрелки?

5.3. МАТЕМАТИЧЕСКИЙ МАЯТНИК

Другим общеизвестным примером колебательной механической системы является «математический» маятник (рис. 5.2). Это идеализированная система, состоящая из частицы или «груза» массой m, прикрепленной к нижнему концу жесткого стержня длиной L с пренебрежимо малой массой, верхний конец которого вращается без трения в точке подвеса. Если груз вывести из положения равновесия и отпустить, то маятник будет совершать ко-лебания в вертикальной плоскости.

Поскольку движение груза происходит по дуге окружности радиуса L с центром в точке O, то положение груза характеризуется длиной дуги или углом θ (см. рис. 5.2). Линейная скорость и ускорение груза при движении по дуге равны

$$v = L\frac{d\theta}{dt}, \tag{5.8}$$

$$a = L\frac{d^2\theta}{dt^2}. \tag{5.9}$$

В отсутствие трения на тело действуют две силы: сила тяжести mg, направленная вертикально вниз, и сила со стороны стержня. Последняя направлена из центра, если $|\theta| < \pi/2$. Поскольку стержень жесткий, то необходимо учесть только компоненту силы mg, направленную по касательной к дуге. Из рис. 5.2 можно понять, что эта компонента равна $mg\sin\theta$ и направлена в сторону уменьшения угла θ. Тогда уравнение движения запишется в виде

$$mL\frac{d^2\theta}{dt^2} = -mg\sin\theta \tag{5.10}$$

или

$$\frac{d^2\theta}{dt^2} = -\frac{g}{L}\sin\theta. \tag{5.11}$$

Уравнение (5.11) является примером нелинейного уравнения, поскольку в него входит $\sin\theta$, а не θ. Большинство нелинейных уравнений не

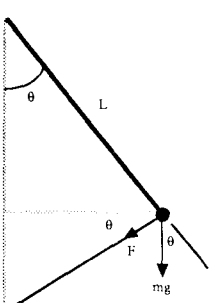

Рис. 5.2. Силы, действующие на математический маятник. Угол θ измеряется относительно вертикальной оси. Он положителен, когда груз находится справа, и отрицателен, когда груз находится слева.

имеет аналитических решений в элементарных функциях и (5.11) не является исключением. Однако если амплитуда колебаний маятника достаточно мала, то $\sin\theta \approx \theta$ и (5.11) можно переписать в виде

$$\frac{d^2\theta}{dt^2} \approx -\frac{g}{L}\theta \tag{5.12}$$

для $\theta \ll 1$. (Напомним, что θ измеряется в радианах.)

Занятие физикой занимательно, в частности, тем, что уравнения, встречающиеся в различных областях (и разных науках), часто оказываются одинаковыми. В результате изучение какого-то явления в одной области позволяет применить полученные знания в другой области. Простой пример такого «перехода» можно увидеть из сравнения уравнений (5.2) и (5.12). Если сопоставить переменные x и θ, то видно, что эти два уравнения имеют одинаковый вид, и можно сразу сделать вывод, что для $\theta \ll 1$ период математического маятника равен

$$T = 2\pi\sqrt{L/g}. \tag{5.13}$$

Один из способов получить представление о движении маятника в случае больших амплитуд колебаний — численно решить уравнение (5.11). Поскольку мы знаем, что численное решение должно удовлетворять условию сохранения полной энергии, то выпишем здесь выражение для нее в явном виде. Потенциальную энергию можно найти, используя следующие соображения. Если стержень отклонить на угол θ, то груз поднимется на высоту $h = L - L\cos\theta$ (см. рис. 5.2). Тогда потенциальную энергию груза в поле тяготения Земли можно записать в виде

$$U = mgh = mgL(1 - \cos\theta), \tag{5.14}$$

где нуль потенциальной энергии соответствует $\theta = 0$. Поскольку кинетическая энергия маятника равна $\frac{1}{2}mv^2 = \frac{1}{2}mL^2(d\theta/dt)^2$, то полная энергия E равна

$$E = \tfrac{1}{2}mL^2(\mathrm{d}\theta/\mathrm{d}t)^2 + mgL(1 - \cos\theta). \tag{5.15}$$

ЗАДАЧА 5.3. Большие колебания математического маятника

а. Для моделирования малых колебаний математического маятника

используйте программу **sho**. Поскольку $\sin\theta \approx \theta$, то ничего менять в ней не надо. Однако необходимо положить $x \to \theta$, $v \to \omega = d\theta/dt$ и $k/m \to g/L$. В данном случае ω является угловой скоростью маятника. В каких единицах измеряется отношение g/L? Возьмите $g/L = 9$, а начальные условия $\theta(t = 0) = 0.1$, $\omega(t = 0) = 0$, и определите период колебаний.

б. Модифицируйте программу так, чтобы промоделировать колебания математического маятника с большой амплитудой, используя уравнение (5.11). Положите $g/L = 9$ и выберите Δt таким, чтобы процедура численного интегрирования давала устойчивое решение, т.е. решение, не отклоняющееся с течением времени от «истинного решения». Проверку устойчивости решения осуществляйте, контролируя величину полной энергии и обеспечивая, чтобы она не отходила от начального значения.

в. Положите $\omega(t = 0) = 0$ и постройте графики $\theta(t)$ и $\omega(t)$ для начальных условий $\theta(t = 0) = 0.1$, 0.2, 0.3, 0.8 и 1.0. (Не забудьте, что угол θ измеряется в радианах и ограничен интервалом $|\theta| < \pi/2$.) Качественно опишите поведение θ и ω. Чему равняются период T и амплитуда θ_m в каждом случае? Постройте график T в зависимости от θ_m и качественно опишите зависимость периода от амплитуды. Сопоставьте результаты, полученные для T в линейном и нелинейном случаях, например в каком случае период больше? Дайте физическое истолкование относительной величины этих периодов T.

*г. Получите несколько картинок фазовой плоскости математического маятника при различных значениях полной энергии. Замкнуты ли фазовые траектории? Зависит ли форма траектории от полной энергии?

5.4. ЗАМЕЧАНИЯ ПО ПРОГРАММИРОВАНИЮ

В этой главе мы познакомимся с некоторыми новыми инструкциями, используемыми для вывода результатов. До сих пор результаты вычислений мы выдавали на экран в табличной или графической форме. В большинстве компьютеров содержимое экрана можно «скинуть» на принтер с помощью «распечатки экрана». Однако часто удобно выводить результаты непосредственно на принтер. В языке True BASIC экран, клавиатура, принтер и дисководы рассматриваются как периферийные «устройст-

ва», доступ к которым компьютер осуществляет с помощью тракта, называемого каналом передачи данных. Чтобы получить доступ к некоторому устройству, необходимо, чтобы был открыт канал доступа, соответствующий этому устройству. Каналам клавиатуры и дисплея присвоены номера #0, причем они открыты всегда. Остальные каналы определяются в инструкции **OPEN**. Например, для доступа к принтеру можно воспользоваться инструкцией

```
OPEN #1: printer ,
```

где номер канала является произвольным. После этого можно вывести на печать значение переменной *pos*, воспользовавшись инструкцией

```
PRINT #1: pos .
```

Последним шагом в этой последовательности инструкций является закрытие данного канала:

```
CLOSE #1
```

Программа **print** открывает канал доступа к принтеру, печатает результат и закрывает канал доступа. Заметим, что номер канала можно передавать в подпрограмму в качестве параметра.

```
PROGRAM print
CALL initial(#1, x)
CALL add(x)
CALL output(#1, x)
END

SUB initial(#1, x)
 OPEN #1: printer
 LET x = 150
END SUB

SUB add(x)
 LET x = x + 200
END SUB
```

```
SUB output(#1,x)
 PRINT #1, using "****.**": x
 CLOSE #1
END SUB
```

Язык True BASIC позволяет создавать мультфильмы, для чего в языке имеется возможность запоминать образы экрана в строковых переменных и выдавать их без каких-либо дополнительных вычислений. Следующая программа, написанная на True BASIC, иллюстрирует использование инструкций **BOX KEEP**, **BOX CLEAR** и **BOX SHOW** для создания иллюзии движения на экране.

```
PROGRAM animation
SET window 1,10,1,10
BOX AREA 1,2,1,2                    ! рисование фигуры и
BOX KEEP 1,2,1,2 in box$ ! ее запоминание в строковой переменной box$
FOR i = 1 to 9 step 0.05
  BOX CLEAR i,i + 1,1,2             ! стирание фигуры
  BOX SHOW box$ at i + 0.05,1       ! рисование фигуры в другом месте
NEXT i
END
```

Чтобы понять качественные различия между линейной и нелинейной возвращающими силами, можно «пронаблюдать» линейный и нелинейный маятники при одних и тех же условиях. Подпрограмма **animation**, входящая в программу **pendula**, создает «картину движения» линейного и нелинейного маятников в двух различных окнах. «Маятники» изображаются в виде окружностей, которые вычерчиваются на экране в точке с координатой, пропорциональной их угловым смещениям. Предыдущие положения маятников стираются, и окружности «движутся» по экрану в горизонтальном направлении.

```
PROGRAM pendula     ! мультфильм о линейном и нелинейном маятниках
CALL              initial(#1,#2,x1,v1,x2,v2,w2,dt,ball$,r)
DO  until  key  input
  CALL linear(x1,v1,w2,dt,x1old)          ! алгоритм Эйлера-Кромера
  CALL        nonlinear(x2,v2,w2,dt,x2old)
  CALL          animation(#1,x1,x1old,ball$,r)
  CALL          animation(#2,x2,x2old,ball$,r)
LOOP
CLOSE #1
CLOSE #2
END

SUB initial(#1,#2,x1,v1,x2,v2,w2,dt,ball$,r)
  INPUT prompt "шаг по времени = ": dt
  LET x1 = 0.5       ! начальное положение (рад) линейного осциллятора
  LET x2 = 0.5       ! начальное положение нелинейного осциллятора
  LET v1 = 0         !начальная угловая скорость (линейный осциллятор)
  LET v2 = 0         ! нелинейный осциллятор
  LET w2 = 9         ! отношение g/L
  LET xmax = x1      ! размер окна
  LET aspect_ratio = 1.5*2     ! поправка для окна в половину экрана
  LET vmax = xmax
  LET hmax = aspect_ratio*xmax
  OPEN #1: screen 0,1,0,0.5      ! нижняя половина экрана
  SET window -hmax,hmax,-vmax,vmax
  PRINT "линейный осциллятор"
  ! вычерчивание окружности
  LET r = 0.1                    ! радиус окружности
  BOX CIRCLE x1 - r,x1 + r,- r,r
  ! запоминание окружности в строковой переменной ball$
  BOX KEEP x1 - r, x1 + r,-r,r in ball$
  OPEN #2: screen 0,1,0.5,1      ! верхняя половина экрана
  SET window -hmax,hmax,-vmax,vmax
  PRINT "нелинейный осциллятор"
  BOX CIRCLE x2 - r,x2 + r,- r,r
END SUB
```

```
SUB animation(#9,theta,old_theta,ball$,r)
  WINDOW #9
  LET x = old_theta
  BOX CLEAR x-r,x+r,-r,r          ! стирание старой окружности
  LET x = theta
  BOX SHOW ball$ at x-r,-r        ! вычерчивание новой окружности
END SUB
```

Подпрограммы **linear** и **nonlinear** не приводятся, поскольку они аналогичны подпрограмме **Euler** из программы **sho**. Единственное различие состоит в том, что предыдущее положение осцилляторов *x1old* и *x2old* нужно передавать в основную программу и подпрограмму **animation**.

***ЗАДАЧА 5.4. Мультипликация линейного и нелинейного осцилляторов**

а. Качественно опишите характер движения маятников. В каком случае движение происходит с большей скоростью?

б. Опишите качественные особенности движения линейного и нелинейного маятников.

в. Напишите программу, повышающую «зрелищность» сеанса наблюдений за счет замены окружности более реалистичным изображением маятника.

5.5. ЗАТУХАЮЩИЕ КОЛЕБАНИЯ

Из опыта известно, что в природе большинство колебаний постепенно уменьшается до тех пор, пока смещение не становится нулевым; такие колебания называются *затухающими*. В качестве примера гармонического осциллятора с затуханием рассмотрим движение блока, показанного на рис. 5.1, с учетом горизонтальной тормозящей силы. Для движения с малыми скоростями в качестве приближения разумно принять, что тормозящая сила пропорциональна первой степени скорости. В этом случае уравнение движения можно записать в виде

$$\frac{d^2x}{dt^2} = -\omega^2 x - \gamma \frac{dx}{dt} , \qquad (5.16)$$

где *коэффициент затухания* γ представляет меру тормозящей силы. Заметим, что тормозящая сила в уравнении (5.16) направлена в сторону, противоположную движению. Как ведет себя $x(t)$, если пренебречь линейным возвращающим членом в уравнении (5.16)? В задаче 5.5 моделируется поведение гармонического осциллятора с затуханием.

ЗАДАЧА 5.5. Гармонический осциллятор с затуханием

а. Учтите в своей программе эффект затухания колебаний и постройте временные зависимости координаты и скорости гармонического осциллятора. Для простоты все расчеты проведите с параметрами $\omega_0^2 = 9$, $x_0 = 1$, $v_0 = 0$.

б. Положите $\gamma = 0.5$ и постройте график функции $x(t)$. Определите период колебаний как время между двумя последовательными максимумами этой функции. Вычислите период и соответствующую угловую частоту и сравните их значения со случаем отсутствия затухания. Будет ли период (частота) больше или меньше? Проведите дополнительные расчеты для случаев $\gamma = 1$, 2 и 3. Увеличивается или уменьшается период (частота) с увеличением коэффициента затухания?

в. Определим амплитуду как максимальное значение функции $x(t)$ на одном периоде колебаний. Вычислите *время релаксации* τ, определяемое как время, за которое амплитуда колебаний уменьшается в $e \approx 2.72$ раз относительно максимального значения. Покажите, что значение τ неизменно в течение всего времени колебаний. Вычислите τ для значений коэффициента γ, рассмотренных в п. «б», и качественно объясните зависимость τ от γ.

г. Постройте график зависимости полной энергии от времени для различных значений коэффициента затухания, приведенных в п. «б». Если энергия уменьшается немонотонно, то объясните причину этого явления.

д. Вычислите средние значения кинетической, потенциальной и полной энергий по полному периоду. Постройте графики этих средних в зависимости от числа периодов. Из-за наличия затухания эти средние уменьшаются с ростом числа периодов. Равномерно ли это уменьшение? Охарактеризуйте временную зависимость этих средних в зависимости от τ для $\gamma = 0.5$ и $\gamma = 1$.

е. Вычислите зависимости $x(t)$ и $v(t)$ для $\gamma = 4$, 5, 6, 7 и 8. Будет ли движение колебательным для всех значений γ? Выберите подходящее рабочее определение равновесия, например $|x| < 0.0001$. Как скоро $x(t)$ достигает равновесия? Осциллятор называется *критически затухающим* при данном ω_0, если γ равно наименьшему значению, при котором осуществляется монотонный переход в равновесие. Для каких значений γ будет иметь место критическое затухание при $\omega_0 = 3$ и $\omega_0 = 2$? Для каждого значения ω_0 вычислите значение γ, при котором система достигает равновесия наиболее быстро.

*ж. Постройте фазовую диаграмму для случаев $\omega_0 = 3$ и $\gamma = 0.5$, 2, 4, 6 и 8. Зависят ли качественные особенности фазовой диаграммы от γ? Если да, то объясните качественные различия.

5.6. ЛИНЕЙНЫЙ ОТКЛИК НА ВНЕШНЮЮ СИЛУ

Каким образом можно определить период колебаний маятника, который еще не двигается? Очевидный способ заключается в том, чтобы подвергнуть систему возмущению, например сместить груз из положения равновесия и наблюдать его движение. В дальнейшем мы обнаружим, что по *отклику* системы на возмущение можно судить о некоторых свойствах невозмущенной системы.

Рассмотрим линейный осциллятор с затуханием, совершающий *вынужденные* колебания под действием вынуждающей силы $F(t)$ дополнительно к линейным возвращающей и тормозящей силам. В этом случае уравнение движения можно записать в виде

$$\frac{d^2x}{dt^2} = -\omega^2 x - \gamma \frac{dx}{dt} + \frac{F(t)}{m}. \tag{5.17}$$

«Отклик» системы принято рассматривать как функцию от смещения x, а не скорости v.

Функция $F(t)$, входящая в уравнение (5.17), зависит от времени произвольным образом. Поскольку многие силы являются гармоническими, рассмотрим сначала силу вида

$$\frac{F(t)}{m} = A_0 \cos \omega t, \tag{5.18}$$

где ω—угловая частота вынуждающей силы. В задаче 5.6 исследуется отклик линейного осциллятора с затуханием на гармоническую силу.

ЗАДАЧА 5.6. Установившийся режим вынужденных колебаний линейного осциллятора с затуханием

а. Модифицируйте программу **sho** так, чтобы учитывалось влияние внешней силы вида (5.18). Введите эту внешнюю силу с помощью внешней функции, которую можно будет легко модифицировать, не меняя основную программу. Например, подпрограмму **Euler** можно составить в следующем виде:

```
SUB Euler(pos, vel, nat_freq2, gamma, ang_freq, t, dt, ncalc)
  DECLARE DEF f                          ! внешняя сила
  FOR icalc = 1 to ncalc
    LET t=t + dt                         ! время (секунды)
    LET accel = -nat_freq2*pos           ! линейная возвращающая сила
    LET accel = accel - gamma*vel        ! затухание
    LET accel = accel + f(t,ang_freq)    ! внешняя сила
    LET vel = vel + accel*dt
    LET pos = pos + vel*dt
  NEXT icalc
END SUB
```

Параметр *nat_freq2* в подпрограмме **Euler** соответствует отношению k/m. Оба параметра *nat_freq2* и угловая частота *ang_freq* внешней силы задаются в подпрограмме **initial**. Внешняя сила определяется как внешняя функция:

```
DEF f(t,w)
   LET A0 = 1              ! амплитуда внешней силы, деленная на массу
   LET f = A0*cos(w*t)
END DEF
```

б. Во всех расчетах п.п. «б»–«д», если не указано особо, используйте параметры $\omega_0 = 3$, $\gamma = 0.5$, $A_0 = 1$ и $\omega = 2$. Из задачи 5.5б известно, что для указанных значений ω_0 и γ поведение системы в отсутствие внешней силы отвечает осциллятору со слабым затуханием. Постройте график функции $x(t)$ при начальных условиях $\{x_0 =$

= 1, $v_0 = 0$}. В чем состоит качественное отличие поведения функции $x(t)$ от случая невозмущенного движения? Чему равны период и угловая частота функции $x(t)$ после нескольких колебаний? Постройте аналогичный график $x(t)$ для начальных условий {$x_0 = 0$, $v_0 = 1$}. Чему равны период и угловая частота функции $x(t)$ после нескольких колебаний? Существует ли у функции $x(t)$ предельная форма, которая не зависит от начальных условий? Зависит ли поведение функции $x(t)$ от начальных условий на коротких временах? Выделите *переходную* часть функции $x(t)$, которая зависит от начальных условий и затухает со временем, и *установившуюся* часть, определяющую поведение функции на больших временах и не зависящую от начальных условий.

в. Вычислите $x(t)$ для $\omega = 1$ и $\omega = 4$. Чему равны в каждом случае период и угловая частота установившегося движения?

г. Вычислите $x(t)$ для $\omega_0 = 4$; остальные параметры возьмите такими же, как в п. «б». Чему равна угловая частота установившихся колебаний? На основании результатов, полученных в п.п. «б»–«г», объясните, какие параметры влияют на частоту установившихся колебаний.

д. Убедитесь в том, что установившееся движение описывается формулой

$$x(t) = A(\omega)\cos(\omega t + \delta), \tag{5.19}$$

где δ — разность фаз между приложенной силой и установившимися колебаниями. Вычислите δ для $\omega_0 = 3$, $\gamma = 0.5$ и значений $\omega = 0$, 1.0, 2.0, 2.2, 2.4, 2.6, 2.8, 3.0, 3.2 и 3.4. Повторите расчет для $\gamma = 1.5$ и постройте график зависимости δ от ω для обеих значений γ. Обсудите качественную зависимость $\delta(\omega)$ для этих двух значений γ.

ЗАДАЧА 5.7. Отклик линейного осциллятора с затуханием

а. Характер вынужденных колебаний гармонического осциллятора на больших временах зависит от частоты приложенной силы. Мерой этих колебаний является максимальное установившееся смещение $A(\omega)$. Приведенные ниже дополнительные инструкции, помещаемые в подпрограмму **Euler**, позволяют вычислять $A(\omega)$ непосредственно. Не за-

будьте передать переменную *amplitude* в основную программу.

```
IF pos > amplitude then
    LET amplitude = pos
    PRINT t, "амплитуда = "; amplitude
END IF
```

Примите начальные условия $\{x_0 = 0,\ v_0 = 0\}$. Вычислите $A(\omega)$ для $\omega = 0,\ 1.0,\ 2.0,\ 2.2,\ 2.4,\ 2.6,\ 2.8,\ 3.0,\ 3.2$ и 3.4 при $\omega_0 = 3$ и $\gamma = 0.5$. Постройте график $A(\omega)$ и качественно объясните поведение этой функции. Если у функции $A(\omega)$ имеется максимум, то определите резонансную угловую частоту ω_m, которая представляет собой частоту в точке максимума A. Близки ли значения резонансной частоты ω_m и собственной частоты ω_0? Оцените численное значение частоты ω_m и сопоставьте количественно это значение с ω_0 и частотой линейного осциллятора с затуханием в отсутствие внешней силы (см. задачу 5.5б).

б. Вычислите A_m — амплитуду в точке $\omega = \omega_m$ — и отношение $\Delta\omega/\omega_m$, где $\Delta\omega$ является «шириной» резонанса. Определите $\Delta\omega$ как ширину интервала частот между точками резонансной амплитудно-частотной характеристики, равными $1/\sqrt{2} \approx 0.707\, A_m$. Положите $\omega_0 = 3$ и рассмотрите $\gamma = 0.1,\ 0.5,\ 1.0$ и 2.0. Качественно опишите зависимости A_m и $\Delta\omega/\omega_m$ от γ. Величина $\Delta\omega/\omega$ пропорциональна $1/Q$, где Q — «добротность» осциллятора.

в. Качественно опишите поведение амплитуды $A(\omega)$ в установившемся режиме в окрестности точки $\omega = 0$ и при частотах $\omega \gg \omega_0$. Почему $A(\omega = 0) > A(\omega)$ при малых ω? Почему $A(\omega) \to 0$ при $\omega \gg \omega_0$?

*г. До сих пор мы находили резонансную частоту ω_m из условия для резонанса установившейся амплитуды. Вычислите среднюю за период колебаний кинетическую энергию в установившемся режиме. Будет ли у кинетической энергии та же резонансная частота, что и у амплитуды? Положите $\omega_0 = 3$, $\gamma = 0.5$ и $A = 1$.

Для многих задач гармоническая вынуждающая сила, рассмотренная в задачах 5.6 и 5.7, не соответствует реальности. Другой вид внешней силы можно обнаружить, наблюдая за тем, как ребенка раскачивают на качелях. Поскольку в этом случае сила отлична от нуля только в течение коротких интервалов времени, такой вид силы называется *им-*

пульсным. В следующей задаче рассматривается отклик на такую силу линейного осциллятора с затуханием.

*ЗАДАЧА 5.8. Отклик на негармоническую внешнюю силу

а. Для простоты предположим, что качели можно промоделировать посредством линейной возвращающей силы и линейного затухания. Модифицируйте программу **sho** так, чтобы учесть внешний периодический импульс. Для простоты положите длительность толчка равной шагу по времени Δt. Поскольку влияние импульса заключается в изменении скорости, учтем влияние внешнего импульса в подпрограмме **Euler** следующим образом:

```
LET ntime = int(t/dt)              ! номер шага по времени
LET vel = vel + accel*dt + impulse(ntime,dt)
```

Оформим импульс как внешнюю функцию, используя функцию mod.

```
DEF impulse(ncount,dt)
  LET Im = 1.0                 ! величина импульса
  LET ang_freq = 2.8           ! угловая частота внешней силы
  LET 2*pi/ang_freq            ! период внешней силы
  LET nperiod = int(T/dt)      ! номер соответствующего шага по времени
  IF mod(ncount,nperiod) < 1 then
    LET impulse = Im
  ELSE
    LET impulse = 0
  END IF
```

Для экономии машинного времени было бы лучше описать угловую частоту внешней силы и параметр *nperiod* в подпрограмме **initial** и передавать последний в функцию **impulse**.

б. Определите амплитуду установившегося движения $A(\omega)$ для различных значений угловой частоты приложенной силы $\omega = 2\pi/T$, при которых импульс не равен нулю. T — время между «толчками». Соответствуют ли полученные результаты вашему опыту по раскачиванию качелей и сравнимым результатам из задачи 5.7?

в. Рассмотрите амплитудную характеристику для внешней силы в ви-

де «полуволны», представляющей собой положительную часть косинуса (рис. 5.3). Вычислите установившуюся амплитуду $A(\omega)$ для линейного осциллятора с затуханием с параметрами $\omega_0 = 3$ и $\gamma = 0.5$ для $\omega = 1.0$, 1.3, 1.4, 1.5, 1.6, 2.5, 3.0 и 3.5. На каких частотах $A(\omega)$ имеет относительные максимумы? Удается ли вам получить какие-нибудь доказательства того, что вынуждающая сила в виде полуволнового косинуса эквивалентна сумме косинусов с разными частотами? Например, имеет ли $A(\omega)$ не один резонанс?

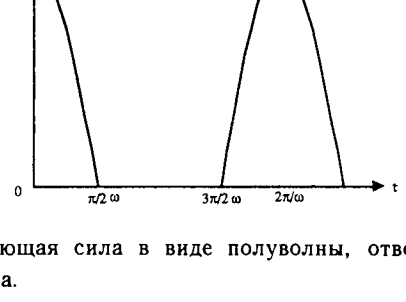

Рис. 5.3. Вынуждающая сила в виде полуволны, отвечающей положительной части косинуса.

5.7. ПРИНЦИП СУПЕРПОЗИЦИИ

До сих пор мы рассматривали отклик системы на одну внешнюю силу. В следующей задаче мы определим отклик линейного осциллятора с затуханием на внешнюю силу, являющуюся суммой гармонических членов.

ЗАДАЧА 5.9. Принцип суперпозиции

а. Положите $\omega_0 = 3$, $\gamma = 0.5$ и $\omega = 2$ и вычислите $x(t)$ для внешней гармонической силы $F_1(t) = \cos \omega t$ и $F_2(t) = 2 \cos \omega t$. Обозначьте соответствующие функции $x_1(t)$ и $x_2(t)$. Как соотносятся установившиеся режимы для $x_2(t)$ и $x_1(t)$? Что вы ожидаете получить в случае $F(t) = 4 \cos \omega t$? Удовлетворяют ли переходные режимы $x(t)$ для разных внешних сил тем же соотношениям, что и установившиеся режимы?

б. Для случая $\omega_0 = 3$ и $\gamma = = 0.5$ вычислите установившийся отклик $x(t)$ на внешнюю силу

$$\frac{1}{m}F(t) = \frac{1}{\pi} + \frac{1}{2}\cos\omega t + \frac{2}{3\pi}\cos 2\omega t - \frac{2}{15\pi}\cos 4\omega t \qquad (5.20)$$

при $\omega = 1.0$. Сравните поведение $x(t)$ с соответствующими результатами для $x(t)$ в случае полуволнового косинуса, рассмотренными в задаче 5.8в.

в. Постройте график функции $F(t)$, заданной в формуле (5.20), для $\omega = 2$. Как соотносится поведение $F(t)$ с полуволновой функцией косинуса (рис. 5.3). На основе полученных результатов попробуйте сформулировать *принцип суперпозиции* для решений линейных уравнений.

В задаче 5.9 мы нашли, что отклик гармонического осциллятора с затуханием на внешнюю вынуждающую силу является линейным. Например, если амплитуду внешней силы увеличить вдвое, то амплитуда установившихся колебаний также удвоится. Такое поведение является следствием линейности уравнения движения. Теперь исследуем отклик нелинейного осциллятора, например осциллятора, у которого возвращающая сила пропоциональна $\sin\theta$, а не θ.

ЗАДАЧА 5.10. Отклик нелинейного осциллятора

Для моделирования нелинейного маятника с гармонической внешней силой и линейным тормозящим членом модифицируйте программу из задачи 5.9. Положите $\omega_0^2 = g/L = 9$, $\gamma = 0.5$ и $\omega = 2$. Определите зависимость амплитуды установившихся колебаний $A(\omega)$ от амплитуды A вынуждающей силы при $A = 0.5$, 1, 2 и 4. Является ли отклик линейным для малых значений A? Применим ли принцип суперпозиции к нелинейным системам?

5.8. КОЛЕБАНИЯ В ЭЛЕКТРИЧЕСКИХ ЦЕПЯХ

Поговорим теперь о некоторых электрических аналогах рассмотренных механических систем. Хотя в этом случае уравнения колебаний имеют тот же масый вид, удобно рассмотреть электрические цепи отдельно, поскольку система обозначений и характер обсуждаемых вопросов несколько отличаются.

Отправной точкой в теории электрических цепей является закон Кирхгофа, который гласит, что сумма падений напряжения на участках

замкнутой цепи равна нулю. Этот закон является следствием сохранения энергии, поскольку падение напряжения представляет собой количество энергии, которая теряется или приобретается в процессе прохождения единичного заряда через элементы цепи. Краткая сводка формул для падения напряжения на каждом элементе цепи представлена в табл. 5.1.

ТАБЛИЦА 5.1. Падение напряжения на основных элементах электрической цепи. Q—заряд (в кулонах) на пластинах конденсатора и I—ток (в амперах)

Элемент	Падение напряжения	Единицы измерения
Резистор	$V_R = IR$	Сопротивление R, омы (Ом)
Конденсатор	$V_C = \frac{Q}{C}$	Емкость C, фарады (Ф)
Катушка индуктивности	$V_L = L\frac{dI}{dt}$	Индуктивность L, генри (Гн)

Представим себе электрическую цепь, состоящую из источника переменного напряжения и соединенных последовательно резистора, индуктивности и конденсатора (рис. 5.4). Соответствующее этой цепи уравнение имеет вид

$$V_L + V_R + V_C = V_s(t). \tag{5.21}$$

В этом уравнении член источника напряжения V_s является *ЭДС*, измеряемой в вольтах. Подставляя соотношения, приведенные в табл. 5.1,

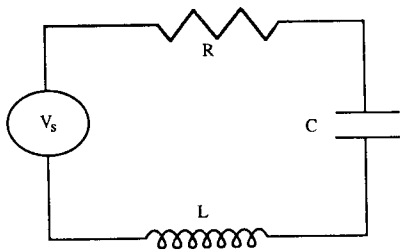

Рис. 5.4. Простая последовательная *RLC*-цепь с источником напряжения V_s.

получим

$$L\frac{d^2Q}{dt^2}+R\frac{dQ}{dt}+\frac{Q}{C}=V_s(t), \tag{5.22}$$

где мы мспользовали определение силы тока $I = dQ/dt$. Видно, что вид уравнения (5.22) для последовательной RLC-цепи идентичен уравнению (5.17) для гармонического осциллятора с затуханием. Аналогии между идеальными электрическими цепями и механическими системами приведены в табл. 5.2.

ТАБЛИЦА 5.2. Аналогии между электрическими и механическими параметрами

Электрическая цепь	Механическая система
Заряд Q	Смещение x
Ток $I = \frac{dQ}{dt}$	Скорость $v = \frac{dx}{dt}$
Падение напряжения	Сила
Индуктивность L	Масса m
Обратная емкость $1/C$	Силовая постоянная k
Сопротивление R	Коэффициент затухания γ

Хотя уравнение (5.22) нам уже знакомо, рассмотрим сначала динамическое поведение RC-цепи, описываемой уравнением

$$RI(t)=R\frac{dQ}{dt}=V_s(t)-\frac{Q}{C}. \tag{5.23}$$

Поскольку сила тока I имеет размерность заряда в единицу времени, то произведение RC должно иметь размерность времени, т.е. секунды. На рис. 5.5 изображены две RC-цепи, описываемые уравнением (5.23). Хотя уравнение (5.23) описывает обе цепи независимо от порядка соединения конденсатора и резистора, выходное напряжение, измеряемое осциллоскопом, будет в каждом случае различным. В задаче 5.11 мы поймем, что эти цепи функционируют как *фильтры*, пропускающие компоненты напряжения определенных частот и отсекающие остальные.

Самым важным достоинством численного моделирования электрических цепей является то, что измерение падения напряжения на элементах цепи не влияет на характеристики самой цепи — мечта любого электротехника! И действительно вычислительные машины часто используются

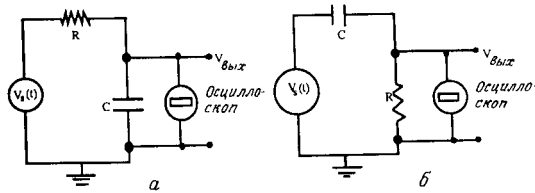

Рис. 5.5. Примеры *RC*-цепей, используемых в качестве фильтров низких и высоких частот. Какая цепь соответствует фильтру низких, а какая высоких частот?

для оптимального проектирования цепей специального назначения. В программе **rc** моделируется *RC*-цепь с источником переменного напряжения вида $V_{вх}(t) = V_0 \cos \omega t$. Временны́е зависимости для источника напряжения и падения напряжения на резисторе демонстрируются в разных окнах. Подпрограмма **screen** вызывает подпрограмму **plot_axis**, которая разрабатывалась в гл. 2. Единственное изменение, которое необходимо в ней сделать, касается инструкции формата

```
PLOT TEXT, at xmax - 0.5*Lγ, γ0: using$("*.***", xmax)
```

Поскольку в программе **rc** широко используется работа с окнами, при написании ее на других языках программирования потребуются значительные переделки.

```
PROGRAM rc
CALL initial(R, tau, V0, w, tmax, dt)
CALL screen(#1, #2, V0, tmax)
CALL scope(#1, #2, R, tau, V0, w, tmax, dt)
CLOSE #1
CLOSE #2
END
```

```
SUB initial(R, tau, V0, w, tmax, dt)
   LET V0 = 1                          ! амплитуда внешнего напряжения
   INPUT prompt "частота внешнего напряжения (Гц) = ":f
   LET w = 2*pi*f                      ! угловая частота
   INPUT prompt "сопротивление (Ом) = ":R
   INPUT prompt "емкость (Ф) = ":C
   INPUT prompt "шаг по времени dt = ": dt
   LET tau = R*C                       ! время релаксации
   LET T = 1/f                         ! период внешней силы
   IF T > tau then
      LET tmax = 2*T
   ELSE
      LET tmax = 2*tau
   END IF
END SUB

SUB screen(#1, #2, V0, tmax)
   OPEN #1: screen 0,1,0,0.5
   LET tmin = 0
   LET Vmin = -V0
   LET title$ = "источник напряжения"
   CALL plot_axis(tmin, tmax, Vmin, V0, title$)
   OPEN #2: screen 0,1,.5,1
   LET title$ = "падение напряжения на резисторе"
   CALL plot_axis(tmin, tmax, Vmin, V0, title$)
END SUB

SUB scope(#1, #2, R, tau, V0, w, tmax, dt)
   ! вычисление падений напряжений и построение графиков
   DECLARE DEF V
   LET Q = 0
   DO while t <= tmax
     LET t = t + dt
     LET I = V(V0, w, t)/R - Q/tau
     LET Q = Q + I*dt
     CALL source_voltage(#1, V0, w, t)
     CALL output_voltage(#2, I, R, t)
   LOOP
END SUB
```

```
DEF V(V0,w,t) = V0*cos(w*t)

SUB source_voltage(#1,V0,w,t)
   DECLARE DEF V
   WINDOW #1
   PLOT LINES: t,V(V0,w,t);
END SUB

SUB output_voltage(#2,I,R,t)          ! падение напряжения на резисторе
   WINDOW #2
   PLOT LINES: t,I*R;
END SUB
```

ЗАДАЧА 5.11. Простые фильтры

а. Воспользуйтесь программой **rc** с параметрами R = 1000 Ом и C = = 1.0 мкФ (10^{-6}фарад). Найдите установившиеся амплитуды падений напряжения на резисторе и конденсаторе в зависимости от угловой частоты ω источника или «входного» напряжения $V_{вх}(t) = V_0 \cos \omega t$. (Для построения графика падения напряжения на конденсаторе программу необходимо модифицировать.) Рассмотрите частоты f = 10, 50, 100, 160, 200, 500, 1000, 5000 и 10000 Гц. Возьмите для f = = 10 Гц значение шага Δt по меньшей мере 0.0001 с. Каково значение шага Δt целесообразно взять для f = 10000 Гц? Зависит ли это от начальных условий?

б. Выходное напряжение зависит от того, где подсоединен осциллоскоп. Чему равно выходное напряжение, измеряемое осциллоскопом, присоединенным, как показано на рис. 5.5,*а*? Постройте график отношения амплитуды выходного напряжения к амплитуде входного напряжения как функцию от ω. Используйте для переменной ω логарифмический масштаб. (Не забудьте, что $\omega = 2\pi f$.) Какие частоты пропускаются этим фильтром? Функционирует ли данная цепь как фильтр высоких или низких частот? Ответьте на аналогичные вопросы для цепи с осциллоскопом, присоединенным, как показано на рис. 5.5,*б*. Воспользуйтесь полученными результатами для объяснения принципа действия фильтров высоких и низких частот. Вычислите значение «критической частоты», при которой амплитуда выходного напряжения падает в $\sqrt{2}$ раз (половинная мощность) относительно

входного значения. Как критическая частота зависит от величины RC?

в. Постройте графики падения напряжения на резисторе и конденсаторе в зависимости от времени. В каждом случае разность фаз ϕ между падением напряжения и напряжением источника можно найти, определив время t_m между соответствующими максимумами этих напряжений. Поскольку ϕ обычно выражается в радианах, мы имеем соотношение $\phi/2\pi = t_m/T$, где T — период колебаний. Чему равны разность фаз ϕ_C между напряжением на конденсаторе и источником и разность фаз ϕ_R между напряжением на резисторе и источником? Зависят ли эти разности фаз от ω? Опережает ли ток напряжение или отстает от него, т.е. наступает ли максимальное значение $V_R(t)$ раньше или позже максимума $V_{вх}(t)$? Чему равна разность фаз между напряжениями на конденсаторе и резисторе? Зависит ли она от ω?

г. Модифицируйте программу **rc** для нахождения установившегося отклика LR-цепи с источником напряжения $V_{вх}(t) = V_0 \cos \omega t$. Положите $R = 100$ Ом, $L = 2 \cdot 10^{-3}$ Гн. Поскольку $L/R = 2 \cdot 10^{-5}$ с, удобно измерять время и частоту в одних и тех же единицах $T_0 = L/R$. Введем $\tau = t/T_m$, $\tilde{\omega} = \omega T_0$ и перепишем уравнение колебаний для LR-цепи в следующем виде:

$$I(\tau) + \frac{dI(\tau)}{d\tau} = \frac{V_0}{R} \cos \tilde{\omega}\tau. \qquad (5.24)$$

Какое нужно взять значение шага $\Delta\tau$? Вычислите установившуюся амплитуду падения напряжения на индуктивности и резисторе для входных частот $f = 10$, 20, 30, 35, 50, 100 и 200 Гц. С помощью полученных результатов объясните, каким образом можно использовать LR-цепь в качестве фильтра высоких или низких частот. Постройте графики падения напряжений на индуктивности и резисторе в зависимости от времени и определите разности фаз ϕ_R и ϕ_L между напряжением на резисторе и источником напряжения и напряжением на индуктивности и источником. Зависят ли эти разности фаз от ω? Опережает ли ток напряжение или отстает от него? Чему равна разность фаз между напряжениями на индуктивности и резисторе? Зависит ли она от ω?

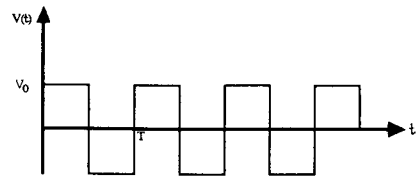

Рис. 5.6. Прямоугольные импульсы с периодом T и амплитудой V_0.

ЗАДАЧА 5.12. Отклик *RC*-цепи на прямоугольный импульс

а. Модифицируйте программу **rc** так, чтобы закон изменения напряжения источника представлял собой прямоугольные импульсы, как показано на рис. 5.6. Положите емкость конденсатора равной 1.0 мкФ, а сопротивление резистора 3000 Ом. По результатам расчета постройте график падения напряжения на конденсаторе в зависимости от времени. Убедитесь в том, что период прямоугольных импульсов достаточно велик, чтобы конденсатор полностью заряжался за один полупериод. Как приблизительно выглядит зависимость $V_C(t)$, пока конденсатор заряжается (разряжается)?

б. Измените входное напряжение на ТТЛ (транзистор-транзисторная логика)-импульсы, как показано на рис. 5.7. Такая серия импульсов используется в качестве часов в цифровых цепях. Чем отличаются функции $V_C(t)$, полученные для случаев ТТЛ-импульсов и периодических прямоугольных импульсов?

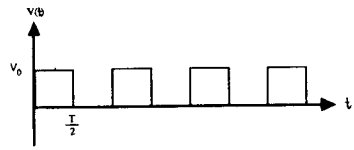

Рис. 5.7. ТТЛ-импульсы с периодом T и амплитудой V_0.

Рассмотрим теперь установившийся режим в последовательной *RLC*-цепи, показанной на рис. 5.4 и описываемой уравнением (5.22). Откликом электрической цепи является зависимость тока, а не заряда на конденсаторе. По аналогии с механической системой

мы уже много знаем о вынужденных колебаний в *RLC*-цепях. Тем не менее в следующих двух задачах мы обнаружим некоторые интересные свойства электрических цепей переменного тока.

ЗАДАЧА 5.13. Отклик *RLC*-цепи

а. Рассмотрите последовательную *RLC*-цепь с параметрами R = 100 Ом, C = 3.0 мкФ и L = 2 мГн. Модифицируйте программу **sho** или программу **rc**, чтобы промоделировать *RLC*–цепь и вычислить падение напряжений на всех трех элементах цепи. Возьмите источник переменного напряжения в виде $V(t) = V_0 \cos \omega t$. Постройте график зависимости тока I от времени. Из графиков, построенных для различных значений ω, определите максимальное значение установившегося тока I_m. Построив зависимость $I_m(\omega)$, получите *резонансную кривую* и вычислите значение ω, при котором она достигает максимума. Это значение ω является *резонансной частотой*.

б. Острота резонансной кривой цепи переменого тока связана с коэффициентом качества или добротностью Q. Не спутайте Q с зарядом конденсатора! Чем острее резонанс, тем больше добротность Q. В схемах настройки любого радиоприемника используются цепи с большим значением добротности Q (а значит, и острым резонансом), что обеспечивает прием в каждый момент времени только одной станции. Определим добротность как $Q = \omega_0/\Delta\omega$, где $\Delta\omega$—ширина частотного интервала между точками резонансной кривой $I_m(\omega)$, имеющими значение, равное 0.707 от ее максимального значения. Вычислите добротность Q для значений R, L и C, приведенных в п. «а». Измените величину R на 10% и вычислите соответствующее процентное изменение добротности Q. Как меняется добротность Q, если L или C изменить на 10%? Понятие добротности Q для механических систем рассматривалось в задаче 5.7б. Заметим, что эти определения добротности Q слегка различаются, но качественно все они имеют один и тот же смысл.

в. Вычислите временну́ю зависимость падения напряжения на каждом элементе цепи приблизительно для пятидесяти частот, лежащих в интервале от 1/10 до 10 резонансной частоты. Постройте графики интервале от 1/10 до 10 резонансной частоты. Постройте графики временны́х зависимостей падения напряжений так, чтобы их можно было использовать в дальнейшем для измерений.

г. Отношение амплитуды синусоидального напряжения источника к амплитуде силы тока называется *импедансом* цепи Z, т.е. $V_m = I_m Z$. Это определение Z является обобщением понятия сопротивления, определяемого для цепи постоянного тока соотношением $V = IR$. С помощью графиков, построенных в п. «в», определите I_m и V_m для различных частот и убедитесь в том, что импеданс равен

$$Z = \sqrt{R^2 + (\omega L - 1/\omega C)^2}. \tag{5.25}$$

При каком значении ω импеданс Z достигает минимума? Заметим, что формула $V = IZ$ справедлива только для максимальных значений I и V, а не для I и V в любой момент времени.

д. Вычислите разность фаз ϕ_R между падением напряжения на резисторе и источником напряжения. Рассмотрите случаи $\omega \ll \omega_0$, $\omega = \omega_0$ и $\omega \gg \omega_0$. Опережает ли в каждом случае ток напряжение или отстает от него, т.е. достигает ли сила тока максимального значения раньше или позже максимума напряжения? Вычислите также разности фаз ϕ_L и ϕ_C и опишите зависимость этих величин от ω. Зависят ли относительные разности фаз между V_C, V_R и V_L от ω?

е. Вычислите амплитуду падения напряжения на индуктивности и конденсаторе на резонансной частоте. Какими будут эти падения напряжения по сравнению с падением напряжения на резисторе и напряжением источника? Сравните также относительные фазы V_C и V_I на резонансной частоте. Объясните, каким образом можно использовать *RLC*-цепь в качестве усилителя входного напряжения.

ЛИТЕРАТУРА

Alfred Bork, Andres Zellweger, Least action via computer, Am. J. Phys. **37**, 386 (1969). Законы Ньютона — не единственно возможная формулировка классической механики.

Least A. Douglas Davis, Classical Mechanics, Academic Press, 1986. Автор кратко описывает обычные Бейсик и Паскаль и приводит простые примеры численных решений уравнений Ньютона. Много внимания уделяется задаче о гармоническом осцилляторе.

Richard P. Feynman, Robert B. Leighton, Matthew Sands, The Feynman Lectures on Physics, Vol.1, Addison-Wesley, 1963. [Имеется перевод: *Фейнман Р., Лейтон Р., Сэндс М.*, Феймановские лекции по фи-

зике. — М.: Мир, 1966.] Гл. 21, 23, 24 и 25 посвящены различным аспектам гармонического движения.

Charles Kittel, Walter D. Knight, Malvin A. Ruderman, Mechanics, 2 ed. revised by *A. Carl Helmholz, Burton J. Moyer,* McGraw-Hill, 1973.

Jerry B. Marion, Classical Dynamics, Academic Press, 1970. Прекрасное обсуждение линейных и нелинейных колебаний.

M. F. McInerney, Computer-aided experiments with the damped harmonic oscillator, Am. J. Phys. **53**, 991 (1985).

J. C. Sprott, Introduction to Modern Electronics, John Wiley & Sons, 1981. Первые пять глав касаются вопросов, обсуждаемых в разд. 5.7.

S. C. Zilio, Measurement and analysis of large-angle pendulum motion, Am. J. Phys. **50**, 450 (1982).

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

Матвеев А. Н., Механика и теория относительности. — М.: Высшая школа, 1986. В гл. 13 изучаются различные аспекты механических колебаний.

Матвеев А. Н., Электричество и магнетизм. — М.: Высшая школа, 1983. В гл. 8 исследуются колебания в цепях переменного тока, в § 55 рассматривается применение *RLC*-цепей в качестве фильтров.

Мандельштам И. Л., Лекции по теории колебаний. — М.: Наука, 1972. Прекрасная книга, в которой автор с присущим ему мастерством излагает все вопросы, затронутые в данной главе. В лекциях 30—32 подробно рассматривается применение *RLC*-цепей в качестве фильтров. Лекция 6 посвящена колебательному контуру (*LC*-цепь).

ПРИЛОЖЕНИЕ 5А. ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ УРАВНЕНИЙ НЬЮТОНА

В заключение приведем несколько наиболее известных конечно-разностных методов решения уравнений движения Ньютона с непрерывными силами. Важно помнить о том, что успешное использование численного метода определяется не только тем, насколько хорошо он приближает производную на каждом шаге, но и тем, насколько хорошо он аппроксимирует интегралы движения, например полную энергию. Множество алгоритмов, используемых в настоящее время, свидетельствует о том, что ни один метод не превосходит по всем параметрам все остальные.

Для упрощения записи рассмотрим одномерное движение частицы и запишем уравнения Ньютона в виде

$$\frac{dv}{dt} = a, \tag{5.26а}$$

$$\frac{dx}{dt} = v. \tag{5.26б}$$

Целью всех конечно-разностных методов является вычисление значений x_{n+1} и v_{n+1} (точка в «фазовом пространстве») в момент времени $t_{n+1} = t_n + \Delta t$. Нам уже известно, что величину шага Δt надо выбирать таким образом, чтобы метод интегрирования порождал устойчивое решение. Один из способов проверки устойчивости метода заключается в контроле величины полной энергии и обеспечении того, чтобы она не отклонялась от начального значения в случае, когда полная энергия сохраняется. Достаточно большое значение шага Δt приводит к несохранению полной энергии и неустойчивым решениям для x_{n+1} и v_{n+1}, т.е. к таким численным решениям, которые все больше отклоняются с течением времени от истинного решения.

Суть многих алгоритмов можно понять, разлагая $v_{n+1} \equiv v(t_n + \Delta t)$ и $x_{n+1} \equiv x(t_n + \Delta t)$ в ряд Тейлора. Запишем

$$v_{n+1} = v_n + a_n \Delta t + O[(\Delta t)^2] \tag{5.27а}$$

и

$$x_{n+1} = x_n + v_n \Delta t + \frac{1}{2} a_n (\Delta t)^2 + O[(\Delta t)^3]. \tag{5.27б}$$

Хорошо известный метод Эйлера эквивалентен сохранению в (5.27) членов $O(\Delta t)$:

$$v_{n+1} = v_n + a_n \Delta t \tag{5.28а}$$

и

$$x_{n+1} = x_n + v_n \Delta t. \tag{5.28б}$$

Поскольку мы удержали в формулах (5.28) члены порядка Δt, то «локальная» погрешность (погрешность на шаге) составляет величину $O(\Delta t)^2$. Однако от шага к шагу погрешности накапливаются, поэтому можно предполагать, что «глобальная» погрешность, представляющая собой суммарную погрешность за рассматриваемый промежуток времени, будет величиной $O(\Delta t)$. Эта оценка погрешности вполне правдоподобна, поскольку число шагов, на которое разбивается временной интервал, пропоционально $1/\Delta t$. Следовательно, порядок глобальной погрешности увеличивается в Δt раз по отношению к локальной погрешности. Поскольку принято говорить, что метод имеет n-й порядок аппроксимации, если его локальная погрешность равна $O((\Delta t)^{n+1})$, то метод Эйлера относится к методам *первого порядка*.

Метод Эйлера является асимметричным, поскольку он продвигает решение на один временной шаг Δt, а использует при этом информацию о производной только в начальной точке интервала! Мы уже убедились в том, что точность метода Эйлера ограничена и зачастую порождаемое им решение неустойчиво. К счастью, как правило, нет необходимости использовать более сложные алгоритмы. Например, ранее мы нашли, что простая модификация метода (5.28), предложенная Кромером и другими авторами, порождает устойчивые решение для колебательных систем. Для полноты изложения повторим алгоритм Эйлера—Кромера или приближение по «последней точке»:

$$v_{n+1} = v_n + a_n \Delta t \tag{5.29а}$$

и

$$x_{n+1} = x_n + v_{n+1} \Delta t. \tag{5.29б}$$

Пожалуй, самый очевидный путь улучшения метода Эйлера состоит в использовании для вычисления нового значения координаты средней на отрезке скорости. Соответствующий метод *средней точки* можно записать в виде

$$v_{n+1} = v_n + a_n \Delta t \tag{5.30а}$$

и

$$x_{n+1} = x_n + \frac{1}{2}(v_{n+1} + v_n)\,\Delta t. \tag{5.30б}$$

Заметим, что если подставить выражение (5.30а) для v_{n+1} в (5.30б), то получим

$$x_{n+1} = x_n + v_n \Delta t + \frac{1}{2} a_n (\Delta t). \tag{5.31}$$

Следовательно, метод средней точки — метод второго порядка точности по координате и первого порядка точности по скорости. Хотя приближение по средней точке дает точные результаты для случая постоянного ускорения, в общем он не приводит к значительно лучшим результатам, чем метод Эйлера. На самом деле оба метода одинаково плохие, поскольку с каждым шагом погрешности увеличиваются.

Метод полушага относится к методам более высокого порядка точности с ограниченной погрешностью. Принимается, что средняя скорость на отрезке равна значению скорости в середине отрезка. Метод полушага можно записать в виде

$$v_{n+1/2} = v_{n-1/2} + a_n \Delta t \tag{5.32а}$$

и

$$x_{n+1} = x_n + v_{n+1/2} \Delta t. \tag{5.32б}$$

Заметим, что метод полушага не является «самостартующим», т.е. формулы (5.36) не позволяют вычислить $v_{1/2}$. Это неудобство можно преодолеть, положив

$$v_{1/2} = v_0 + \frac{1}{2} a_0 \Delta t. \tag{5.32в}$$

Поскольку формулы (5.32) можно повторять до бесконечности, то метод полушага получил широкое распространение в учебной литературе (см., например, книгу Фейнмана и др. и Эйсберга и Лернера).

Один из наиболее известных «несносовых» алгоритмов высокого порядка приписывается Верле. По аналогии с (5.27) запишем разложение в ряд Тейлора для x_{n-1}:

$$x_{n-1} = x_n - v_n\Delta t + \frac{1}{2}a_n(\Delta t). \tag{5.33}$$

Если сложить формулы интегрирования вперед и назад [выражения (5.27) и (5.33) соответственно], то получим

$$x_{n+1} + x_{n-1} = 2x_n + a_n(\Delta t) + O[(\Delta t)] \tag{5.34}$$

или

$$x_{n+1} = 2x_n - x_{n-1} + a_n(\Delta t). \tag{5.35а}$$

Аналогично вычитание разложений в ряд Тейлора для x_{n+1} и x_{n-1} дает

$$v_n = \frac{x_{n+1} - x_{n-1}}{2\Delta t}. \tag{5.35б}$$

Заметим, что связанная с *алгоритмом Верле* (5.35) глобальная погрешность имеет третий порядок для координаты и второй порядок для скорости. Однако скорость не участвует в интегрировании уравнений движения. В литературе по численному анализу алгоритм Верле называется «неявной симметричной разностной схемой».

Недостатком алгоритма Верле является то, что он не самостартующий, и необходимо использовать другой алгоритм для получения нескольких первых точек фазового пространства. Еще один недостаток состоит в том, что новая скорость находится по формуле (5.35б) вычитанием близких по величине чисел. Как говорилось в гл. 2, такая операция обусловливает потерю значащих цифр и может приводить к значительному росту погрешности округления.

Менее известной, но математически эквивалентной версией алгоритма Верле является схема

$$x_{n+1} = x_n + v_n\Delta t + \frac{1}{2}a_n(\Delta t) \tag{5.36а}$$

и

$$v_{n+1} = v_n + \frac{1}{2}(a_{n+1} + a_n)\Delta t. \tag{5.36б}$$

Видно, что схема (5.36), называемая *скоростной* формой алгоритма Верле, является самостартующей и не приводит к накоплению погрешности округления. Формулы (5.36) можно «вывести» из (5.35) следую-

щим образом.. Сначала прибавим и вычтем из обеих частей равенства (5.35а) по $(1/2)x_{n+1}$ и запишем

$$x_{n+1} = x_n + \frac{1}{2}(x_{n+1} - x_{n-1}) - \frac{1}{2}x_{n-1} + \frac{1}{2}x_{n-1} + x_n + a_n(\Delta t)^2 =$$

$$= x_n + v_n\Delta t - \frac{1}{2}(x_{n+1} - 2x_n + x_n) + a_n(\Delta t)^2 \qquad (5.37)$$

Здесь мы воспользовались выражением (5.35б). Из (5.35а) найдем a_n для метода Верле:

$$a_n = \frac{x_{n+1} - 2x_n + x_{n-1}}{(\Delta t)^2}. \qquad (5.38)$$

Легко видеть, что подстановка (5.38) в выражение (5.37) приводит к (5.36а). В том же духе перепишем (5.35а) для v_{n+1}:

$$v_{n+1} = \frac{x_{n+2} - x_n}{2\Delta t}. \qquad (5.39)$$

Теперь перепишем формулу (5.35а) для x_{n+2} и подставим полученный результат в (5.39). Имеем

$$v_{n+1} = \frac{x_{n+1} + v_{n+1}\Delta t + \frac{1}{2}a_{n+1}(\Delta t)^2 - x_n}{\Delta t}. \qquad (5.40)$$

Затем, используя выражение (5.35а) для x_{n+1}, повторим эту процедуру и подставим x_{n+1} в (5.40); после несложных выкладок получаем требуемый результат (5.36б).

Другой полезный алгоритм, в котором нет накопления погрешности округления, присущего алгоритму Верле, принадлежит Биману и Шофилду. Запишем алгоритм *Бимана* в следующем виде:

$$x_{n+1} = x_n + v_n\Delta t + \frac{1}{6}(4a_n - a_{n-1})(\Delta t)^2 \qquad (5.41а)$$

и

$$v_{n+1} = v_n + \frac{1}{6}(2a_{n+1} + 5a_n - a_{n-1})\Delta t. \qquad (5.41б)$$

Заметим, что точность расчета траекторий по схеме (5.41) не выше, чем в алгоритме Верле. Ее преимущество заключается, пожалуй, в том,

что обычно она лучше сохраняет энергию. Однако алгоритм Бимана не самостартующий. Алгоритм Бимана и алгоритм Верле в скоростной форме использованы в программе **Beeman** (см. задачу 5.15).

Завершим наше обсуждение кратким изложением двух методов, которые обычно приводятся в учебниках по численному анализу. Один пример метода *предиктор-корректор* определяется следующим образом. Сначала мы *предсказываем* (predictor) новое значение координаты:

$$\textit{предиктор}: \quad \bar{x}_{n+1} = x_{n-1} + 2v_n\Delta t. \tag{5.42а}$$

Предсказанное значение координаты позволяет определить ускорение $\bar{a}_{n+1}$. Затем, используя $\bar{a}_{n+1}$, получаем *скорректированные* (corrector) значения v_{n+1} и x_{n+1}:

$$\textit{корректор}: \quad v_{n+1} = v_n + \frac{1}{2}(\bar{a}_{n+1} + a_n)\Delta t$$

$$x_{n+1} = x_n + \frac{1}{2}(v_{n+1} + v_n)\Delta t. \tag{5.42б}$$

Скорректированное значение x_{n+1} используется для вычисления нового предсказанного значения a_{n+1} и, значит, новых предсказанных значений v_{n+1} и x_{n+1}. Эта процедура повторяется до тех пор, пока предсказанное и скорректированное значения x_{n+1} отличаются меньше чем на заданную величину. Данный метод можно обобщить на схемы более высокого порядка, которые связывают между собой не только x_{n+1}, x_n и v_n, но также и значения v_{n-1} и v_{n-2}. Заметим, что метод предиктора-корректора не является самостартующим.

Для объяснения метода Рунге—Кутты рассмотрим сначала решение дифференциального уравнения первого порядка

$$\frac{dy}{dx} = f(x, y). \tag{5.43}$$

Метод Рунге—Кутты *второго порядка* для решения уравнения (5.43) можно, пользуясь стандартными обозначениями, записать в следующем виде:

$$k_1 = f(x_n, y_n)\Delta x,$$

$$k_2 = f(x_n + \tfrac{1}{2}\Delta x, y_n + \tfrac{1}{2}k_1)\Delta x, \qquad (5.44)$$

$$y_{n+1} = y_n + k_2 + O[(\Delta x)^3].$$

Смысл формул (5.44) состоит в следующем. В методе Эйлера предполагается, что для экстраполяции в следующую точку можно использовать наклон кривой $f(x_n, y_n)$ в точке (x_n, y_n), т.е. $y_{n+1} = y_n + f(x_n, y_n) \times \Delta x$. Однако можно повысить точность оценки наклона, если методом Эйлера провести экстраполяцию в *срединную* точку отрезка, а затем использовать центральную производную на всем отрезке. Отсюда оценка наклона в методе Рунге—Кутты равна $f(x_n + \frac{1}{2}\Delta x, y^*)$, где $y^* = y_n + \frac{1}{2}f(x_n, y)\Delta x$ [см. (5.48)].

Применение метода Рунге—Кутты к уравнениям движения Ньютона (5.26) дает

$$\begin{aligned} k_{1,x} &= v_n \Delta t, \\ k_{1,v} &= a_n \Delta t, \\ k_{2,x} &= (v_n + \tfrac{1}{2}k_{1,v})\Delta t, \\ k_{2,v} &= a(x_n + \tfrac{1}{2}k_{1,x})\Delta t, \\ x_{n+1} &= x_n + k_{2,x}, \\ v_{n+1} &= v_n + k_{2,v}. \end{aligned} \qquad (5.45)$$

Поскольку методы Рунге—Кутты являются самостартующими, то их часто используют для вычисления нескольких первых шагов для несамостартующих алгоритмов.

Как уже подчеркивалось, не следует отдавать предпочтение одному какому-нибудь алгоритму, даже если учебники вроде нашего советуют это. Успехи в компьютерной технологии в настоящее время позволяют легко экспериментировать с различными алгоритмами для разнообразных динамических систем. В следующей задаче вы приобретете некоторый опыт в сравнении алгоритмов.

ЗАДАЧА 5.15. Сравнение алгоритмов

а. Рассмотрите частицу единичной массы, движущуюся в потенциале Морза

$$V(x) = e^{-2x} - 2e^{-x} \tag{5.46}$$

с полной энергией $E = \frac{1}{2}p^2 + V(x) < 0$. Сила, действующая на частицу, равна

$$F(x) = -\frac{dV}{dx} = 2e^{-x}[e^{-x} - 1]. \tag{5.47}$$

Постройте графики функций $V(x)$ и $F(x)$. Где $V(x)$ достигает минимума? Чему равна полная энергия при начальных условиях $x_0 = 2$ и $v_0 = 0$? Какой характер движения вы ожидаете?

б. Используйте программу **Beeman** и сравните алгоритмы Эйлера—Кромера, Верле в скоростной форме и Бимана, вычисляя $x(t)$, $v(t)$ и E_n, где E_n—полная энергия после n-го временно́го шага. Одной из мер погрешности является разность $\Delta E = \langle E \rangle - E_0$, где E_0—начальная энергия, а $\langle E \rangle$—средняя энергия, равная

$$\langle E \rangle = \frac{1}{n+1}\sum_{i=0}^{n} E_i \,. \tag{5.48}$$

Другой мерой погрешности служит величина δE, определяемая выражением

$$\delta E = \sqrt{\langle E^2 \rangle - \langle E \rangle^2} \,, \tag{5.49}$$

где

$$\langle E^2 \rangle = \frac{1}{n+1}\sum_{i=0}^{n} E_i^2. \tag{5.50}$$

Вычислите ΔE и δE для значений $\Delta t = 0.2$ и 0.1. Если метод имеет второй порядок точности, то данное относительное изменение шага Δt должно уменьшить глобальную погрешность аппроксимации приблизительно в 4 раза. Отсутствие такого уменьшения будет свидетель-

ствовать о том, что, возможно, существенную роль играют погрешности округления. Чтобы увидеть влияние погрешностей округления, повысьте точность вычислений — либо воспользовавшись переменными удвоенной точности в Фортране, либо прибегнув к компьютеру с большей точностью представления чисел. Если движение периодическое, то вычислите период.

в. Сравните полученные результаты для функции $x(t)$ с аналитическим результатом

$$x(t) = \ln(\alpha \cos \omega t + \beta \sin \omega t - E_0^{-1}), \tag{5.51}$$

где

$$\alpha = \exp(x_0) + E_0^{-1},$$

$$\beta = \omega^{-1} v_0 \exp(x_0),$$

$$\omega = (2|E_0|)^{1/2}.$$

г. Повторите вычисления п. «а» с начальными условиями $x_0 = 3$ и $v_0 = 1$. Является ли движение периодическим? Какой алгоритм предпочтительнее в этом случае?

В программе **Beeman** моделируется осциллятор Морза с помощью алгоритма Бимана. Поскольку этот алгоритм не самостартующий, то для вычисления значений x_1, v_1 и a_1 используется скоростная форма алгоритма Верле.

```
PROGRAM Beeman                    ! моделирование осциллятора Морза
CALL initial(x, v, aold, dt, dt2, nmax)
CALL energy(x, v, ecum, e2cum)    ! вычисление начальной энергии
CALL Verlet(x, v, a, aold, dt, dt2)
CALL energy(x, v, ecum, e2cum)
LET n = 1
DO while n < nmax
  LET n = n + 1                   ! число шагов
  CALL Beeman(x, v, a, aold, dt, dt2)
  ! вычисление полной энергии после каждого шага по времени
  CALL energy(x, v, ecum, e2cum)
LOOP
CALL output(ecum, e2cum, n)
END

SUB initial(x, v, aold, dt, dt2, nmax)
 DECLARE DEF f                    ! сила задается как внешняя функция
 LET x = 2
 LET v = 0
 LET aold = f(x)
 INPUT prompt "шаг по времени (c) = ":dt
 LET dt2 = dt*dt
 INPUT prompt "длительность = ":tmax
 LET nmax = tmax/dt
END SUB

SUB Verlet(x, v, a, aold, dt, dt2)
 DECLARE DEF f
 LET x = x + v*dt + 0.5*aold*dt2
 LET a = f(x)
 LET v = v + 0.5*(a + aold)*dt
END SUB
```

```
SUB Beeman(x, v, a, aold, dt, dt2)
 DECLARE DEF f
 LET x = x + v*dt + (4*a - aold)*dt2/6
 LET anew = f(x)                          ! значение на (n+1)-м шаге
 LET v = v + (2*anew + 5*a - aold)*dt/6
 LET aold = a                             ! значение на (n-1)-м шаге
 LET a = anew                             ! значение на n-м шаге
END SUB

DEF f(x)
 LET e = exp(-x)
 LET f = 2*e*(e - 1)
END DEF

SUB energy(x, v, ecum, e2cum)
 LET KE = 0.5*v*v
 LET e = exp(-x)
 LET PE = e*(e - 2)
 LET etot = KE + PE
 LET ecum = ecum + etot
 LET e2cum = e2cum + etot*etot
END SUB

SUB output(ecum, e2cum, n)
 LET n = n + 1                            ! вычисление начального значения
 LET ebar = ecum/n
 PRINT "средняя энергия = "; ebar
 LET sigma2 = e2cum/n - ebar*ebar
 PRINT "sigma = "; sqr(sigma2)
END SUB
```

ЛИТЕРАТУРА

F. S. Acton, Numerical Methods That Work, Harper and Row, 1970, Гл. 5.

R. P. Feynman, R. B. Leighton, M. Sands, The Feynman Lectures on Physics, Vol. 1, Addison-Wesley, 1963. [Имеется перевод: *Фейнман Р., Лейтон Р., Сэндс М.*, Феймановские лекции по физике. — М.: Мир, 1966.] См. гл. 9.

W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling,

Numerical Recipes, Cambridge University Press, 1986. [Готовится перевод: Пресс У., Фланнери Б., Тьюкольски С. и др. Численные рецепты. — М.: Мир, 1990.] В гл. 15 обсуждается интегрирование обыкновенных дифференциальных уравнений.

ДИНАМИКА СИСТЕМ МНОГИХ ЧАСТИЦ

6

Моделируется динамическое поведение систем многих частиц и наблюдаются их качественные свойства. Вводятся некоторые основные понятия равновесной статистической механики и кинетической теории.

6.1. ВВЕДЕНИЕ

До сих пор мы изучали динамику систем, состоящих только из нескольких частиц. Однако на самом деле многие системы, такие как газы, жидкости и твердые тела, состоят из большого числа взаимодействующих друг с другом частиц. В качестве иллюстрации рассмотрим две чашки кофе, сваренного в одинаковых условиях. В каждой чашке содержится примерно $10^{23}-10^{25}$ молекул, движение которых с хорошей точностью подчиняется законам классической физики. Хотя межмолекулярные силы порождают сложные траектории каждой молекулы, наблюдаемые свойства кофе в каждом сосуде неразличимы и их сравнительно легко описать. Известно, например, что температура кофе, если его оставить в чашке, достигает комнатной и с течением времени больше не меняется. Как связана температура кофе с траекториями отдельных молекул? Почему она не зависит от времени, даже если траектории отдельных молекул непрерывно меняются?

Этот пример с чашкой кофе ставит нас перед проблемой: как можно, исходя из известных межмолекулярных взаимодействий, понять наблюдаемое поведение сложной многочастичной системы? Самый очевидный подход — решить эту задачу в лоб, моделируя на компьютере саму задачу многих частиц. Можно себе представить, что на каком-нибудь суперкомпьютере будущего будут решаться микроскопические уравнения движения для 10^{25} взаимодействующих между собой частиц. На самом деле этот подход, называемый методом *молекулярной динамики*, применили к «небольшим» системам многих частиц, насчитывающим обычно от нескольких сотен до нескольких тысяч частиц, и он уже много дал для понимания наблюдаемых свойств газов, жидкостей и твердых тел. Однако детальное знание траекторий 10^4 или даже 10^{25} частиц ничего не даст, если мы не знаем, какие именно вопросы требуют выяснения. Какие основные свойства и закономерности проявляют системы многих частиц? Какие параметры нужно использовать для описания таких систем? К подобным вопросам обращается статистическая механика, и многие ее представления нашли отражение в этой главе. Тем не менее единственное, что требуется для работы над этой главой, это умение численно решать уравнения Ньютона, чем мы уже занимались, и некоторое знакомство с кинетической теорией.

6.2. ПОТЕНЦИАЛ МЕЖМОЛЕКУЛЯРНОГО ВЗАИМОДЕЙСТВИЯ

Первым делом нам необходимо определить *модель* системы, которую мы желаем моделировать. Поскольку мы хотим понять качественные свойства систем многих частиц, пойдем на упрощение задачи, предполагая, что динамику можно считать классической, а молекулы — химически инертными шариками. Мы предполагаем также, что сила взаимодействия любых двух молекул зависит только от расстояния между ними. В этом случае полная потенциальная энергия U определяется суммой двухчастичных взаимодействий:

$$U = V(r_{12}) + V(r_{13}) + \ldots + V(r_{23}) + \ldots = \sum_{i<j=1}^{N} V(r_{ij}), \tag{6.1}$$

где $V(r_{ij})$ зависит только от абсолютной величины расстояния r_{ij} между частицами i и j. Парное взаимодействие вида (6.1) соответствует «простым» жидкостям, например жидкому аргону.

В принципе форму $V(r)$ для электрически нейтральных атомов можно построить путем детального расчета, базирующегося на основных законах квантовой механики. Такой расчет очень труден и, кроме того, обычно бывает вполне достаточно в качестве $V(r)$ выбрать простую феноменологическую формулу. Наиболее важными особенностями $V(r)$ для простых жидкостей является сильное отталкивание для малых r и слабое притяжение на больших расстояниях. Отталкивание при малых r обусловлено правилом запрета. Иначе говоря, если электронные облака двух атомов перекрываются, некоторые электроны должны увеличить свою кинетическую энергию, чтобы находиться в различных квантовых состояниях. Суммарный эффект выражается в отталкивании между электронами, называемом *отталкиванием кора*. Слабое притяжение при больших r обусловлено главным образом взаимной поляризацией каждого атома; результирующая сила притяжения называется *силой Ван-дер-Ваальса*.

Одной из наиболее употребительных феноменологических формул для $V(r)$ является потенциал Леннарда — Джонса

$$V(r) = 4\varepsilon[(\frac{\sigma}{r})^{12} - (\frac{\sigma}{r})^{6}]. \tag{6.2}$$

График потенциала Леннарда—Джонса показан на рис. 6.1. Хотя зависимость r^{-6} в формуле (6.2) получена теоретически, зависимость r^{-12}

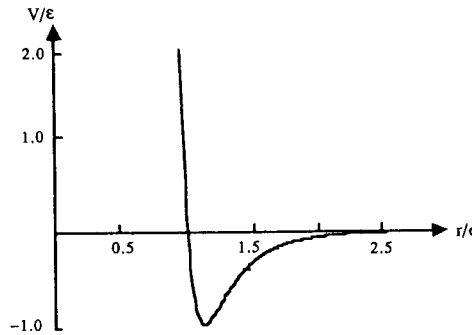

Рис. 6.1. График потенциала Леннарда—Джонса. Отметим, что r измеряется в единицах σ и $V(r)$—в единицах ε.

выбрана только из соображений удобства. Потенциал Леннарда—Джонса параметризуется «длиной» σ и «энергией» ε. Заметим, что при $r = \sigma$ $V(r) = 0$. Параметр ε представляет собой глубину потенциальной ямы в точке минимума $V(r)$; этот минимум расположен при расстоянии $r = 2^{1/6}\sigma$. Заметим, что данный потенциал является «короткодействующим» и $V(r)$ для $r > 2.5\sigma$ по существу равен нулю.

Удобно выражать длины, энергию и массу в единицах σ, ε и m, где m—масса частиц. Мы измеряем скорости в единицах $(\varepsilon/m)^{1/2}$, а время—в единицах $\tau = (m\sigma^2\varepsilon)^{1/2}$. Для жидкого аргона параметры ε и σ потенциала Леннарда—Джонса составляют $\varepsilon/k_B = 119.8$ K и $\sigma = 3.405$ Å. Масса атома аргона равна $6.69 \cdot 10^{-23}$ г, и отсюда $\tau = 1.82 \times 10^{-12}$ с. Во избежание путаницы мы отмечаем все безразмерные или приведенные величины звездочкой. Например, приведенная двумерная плотность определяется соотношением $\rho^* = \rho/\sigma^2$.

6.3. ЧИСЛЕННЫЙ АЛГОРИТМ

Теперь, когда у нас есть четко описанная модель системы многих частиц, необходимо познакомиться с каким-нибудь методом численного интегрирования для расчета траекторий каждой частицы. Уже в своих

первых расчетах по моделированию уравнений движения Ньютона мы пришли к заключению, что устойчивость численного решения можно проконтролировать, следя за полной энергией и убеждаясь, что она не ушла от своего первоначального значения. Как можно было предполагать, алгоритмы Эйлера и Эйлера—Кромера не могут обеспечить сохранение энергии на временах, рассматриваемых при моделировании молекулярной динамики. К счастью, обычно нам не приходится привлекать сложный алгоритм. Интуитивно особенно привлекательным кажется алгоритм, определяемый формулами

$$x_{n+1} = x_n + v_n \Delta t + \frac{1}{2} a_n (\Delta t)^2, \tag{6.3а}$$

$$v_{n+1} = v_n + \frac{1}{2}(a_{n+1} + a_n)\Delta t. \tag{6.3б}$$

Для упрощения обозначений мы записали этот алгоритм только для одной компоненты движения частицы. Алгоритм в виде (6.3) называется *алгоритмом Верле в скоростной форме*, и мы обсудим его в приложении 5А. В литературе по молекулярной динамике широко используется эквивалентная, но другая форма записи алгоритма Верле. Поскольку новая координата x_{n+1} вычисляется с использованием не только скорости v_n, но и ускорения a_n, алгоритм Верле обладает «более высоким порядком» по Δt, чем алгоритмы Эйлера и Эйлера—Кромера. Новая координата используется для нахождения нового ускорения a_{n+1}, которое вместе с a_n используется для получения новой скорости v_{n+1}.

6.4. КРАЕВЫЕ УСЛОВИЯ

Полезное моделирование должно включать в себя все характерные особенности рассматриваемой физической системы. Напомним, что конечной целью наших модельных расчетов является получение оценок поведения макроскопических систем, т.е. систем, содержащих порядка $N \approx 10^{23}$—10^{25} частиц. Рассмотрим сферический резервуар с водой. Доля молекул воды вблизи стенок пропорциональна отношению поверхности к объему $(4\pi R^2)/(4\pi R^3/3)$. Поскольку $N = \rho(4/3\pi R^3)$, где ρ—плотность, доля частиц вблизи стенок пропорциональна $N^{2/3}/N = N^{-1/3}$, что при $N \approx$ $\approx 10^{23}$ пренебрежимо мало. По сравнению с этим количество частиц, которое можно изучать в моделях молекулярной динамики, составляет обычно 10^2—10^4, и доля частиц вблизи стенок не мала. В результате мы не можем провести моделирование макроскопической системы, поме-

щая частицы в резервуар с жесткими стенками. Кроме того, если частица отражается от жесткой стенки, ее положение, а значит, и потенциальная энергия взаимодействия изменяются без какого-либо изменения в ее кинетической энергии. Отсюда присутствие жестких стенок означало бы, что полная энергия системы сохраняется.

Один из способов минимизировать поверхностные эффекты и более точно промоделировать свойства макроскопической системы заключается в использовании *периодических краевых условий*. Реализация периодических краевых условий для короткодействующих взаимодействий, таких как потенциал Леннарда—Джонса, хорошо знакома всем играющим в видеоигры. Рассмотрим сначала одномерный «ящик», содержащий N частиц, движение которых ограничено отрезком прямой длиной L. Концы отрезка прямой играют роль «стенок». Использование периодических краевых условий эквивалентно сворачиванию этого отрезка в кольцо (рис. 6.2). Заметим, что, поскольку расстоянию между частицами соответствует отрезок дуги, максимальное расстояние равно $L/2$.

В двумерном случае мы можем представить себе ящик, у которого противоположные ребра соединены так, что ящик превращается в поверхность тора (форма тороидальной камеры). Поэтому, если частица пересекает ребро ящика, она входит заново в противолежащее ребро. Заметим, что максимальное расстояние между частицами в x- и y-направлениях равно $L/2$, а не L.

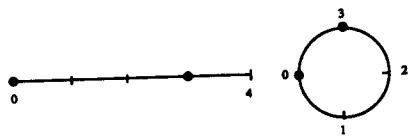

Рис. 6.2. *а*—две частицы в точках $x = 0$ и $x = 3$ на отрезке длиной $L = 4$; расстояние между частицами равно 3; *б*—отрезок преобразован в окружность; кратчайшее расстояние между обеими частицами на окружности равно 1.

То же самое можно представлять и по-другому, как проиллюстрировано на рис. 6.3. Предположим, что частицы 1 и 2 находятся в центральной клетке. Клетка окружена периодически повторяющимися собст-

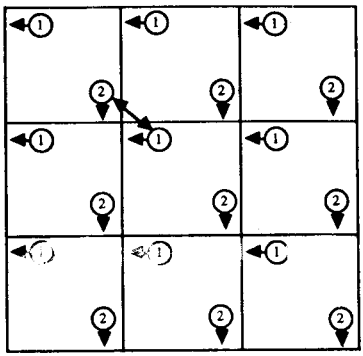

Рис. 6.3. Пример периодических краевых условий в двумерном случае. Обратите внимание на то, что частица 1 собирается покинуть левую грань центральной клетки и войти в центральную клетку справа. Правило ближайшей частицы означает, что расстояние между частицами 1 и 2 определяется жирной линией.

венными копиями; каждая копия клетки содержит обе частицы в тех же относительных положениях. Когда частица влетает в центральную клетку или вылетает из нее с одной стороны, это перемещение сопровождается одновременным вылетом или влетом копии этой частицы в соседнюю клетку с протовоположной стороны. Вследствие использования периодических краевых условий частица 1 взаимодействует с частицей 2 в центральной клетке и со всеми периодическими копиями частицы 2. Однако для короткодействующих взаимодействий мы можем принять правило *ближайшей частицы*. Это правило означает, что частица 1 из центральной клетки взаимодействует только с ближайшим экземпляром частицы 2; взаимодействие полагается равным нулю, если расстояние до копии больше $L/2$ (см. рис. 6.3). Поскольку из данного правила следует, что мы можем визуально представлять себе центральную клетку в виде тора, это использование периодических краевых условий вместе с правилом ближайшей частицы было бы точнее назвать *тороидальными* краевыми условиями. Однако мы верны принятым традициям и называем эти условия периодическими краевыми условиями. Отметим, что использование периодических краевых условий означает, что все узлы ящика эквивалентны.

6.5. ПРОГРАММА МОЛЕКУЛЯРНОЙ ДИНАМИКИ

Теперь у нас все готово для разработки программы молекулярной динамики с прослеживанием на экране траекторий частиц. Будем рассматривать двумерную систему, поскольку в этом случае результаты легче визуализировать и вычисления не занимают очень много времени центрального процессора (ЦП). Структура программы **md** имеет следующий вид:

```
PROGRAM md
DIM x(30), y(30)
DIM vx(30), vy(30), ax(30), ay(30)
CALL initial(x, y, vx, vy, N, Lx, Ly, dt, dt2, nsnap)
CALL screen(x, y, N, Lx, Ly, ball$, r)
CALL accel(x, y, ax, ay, N, Lx, Ly, pe0)
DO
  FOR isnap = 1 to nsnap              ! шаги по времени между снимками
    CALL Verlet(x, y, vx, vy, ax, ay, N, Lx, Ly, dt, dt2, virial, xflux, yflux, pe, ke)
  NEXT isnap
  LET itime = itime + nsnap           ! номер шага по времени
  CALL snapshot(x, y, N, Lx, Ly, ball$, r)          ! вывод координат на экран
  CALL output(N, Lx, Ly, virial, xflux, yflux, pe, ke, nsnap, itime)
LOOP until key input
END
```

Заметим, что *x*- и *y*-компоненты координат, скоростей и ускорений представлены массивами. *Lx* и *Ly* равны горизонтальной и вертикальной длинам прямоугольного резервуара. В переменных *ke* и *pe* накапливаются суммы для кинетической и потенциальной энергии; величины *xflux*, *yfluy* и *virial* понадобятся для вычисления давления. Характер этих величин будет рассмотрен позднее.

Поскольку наша система является детерминированной, характер движения определяется начальными условиями. Правильно выбрать начальные условия труднее, чем могло бы показаться на первый взгляд. Как, например, выбрать начальную конфигурацию (совокупность координат и скоростей), чтобы она отвечала жидкости с искомой температурой? Мы откладываем обсуждение таких вопросов до разд. 6.6 и рассмотрим сначала характер эволюции системы из произвольных начальных состояний.

Один из возможных вариантов начальных условий выглядит так: частицы помещают в узлах прямоугольной сетки и выбирают x- и y-компоненты скоростей случайным образом. В большинстве языков программирования имеется функция, которая генерирует последовательность «случайных чисел» на отрезке [0, 1]. Поскольку ЭВМ детерминирована, она не может вычислять последовательности действительно случайных чисел. Тем не менее компьютер может генерировать числа в совершенно неочевидном порядке, и, что касается нашей задачи, это различие не имеет значения. Разговор о последовательностях случайных чисел пойдет в гл. 11. В языке True BASIC мы имеем возможность использовать функцию **rnd**, которая генерирует случайные числа в диапазоне $0 \leq rnd < 1$. Случайные значения v_x на отрезке $[-v_{max}, v_{min}]$ формируются с помощью инструкции

```
LET vx(i) = vmax*(2*rnd-1)
```

Количество частиц, линейные размеры системы и начальные координаты и скорости частиц задаются в подпрограмме **initial**. Заметим, что Lx и Ly измеряются в единицах σ—параметра потенциала Леннарда—Джонса. Поскольку скорости выбираются случайно, их следует подправить, имея в виду, что начальный полный импульс в x- и y-направлении может просто не получиться равным нулю. Сетка в подпрограмме **initial** выбирается так, чтобы в начальный момент $N = 12$ частиц находились в левой половине ящика.

```
SUB initial(x(),y(),vx(),vy(),N,Lx,Ly,dt,dt2,nsnap)
   LET N = 12                              ! число частиц
   INPUT prompt "Lx и Ly = ": Lx, Ly
   INPUT prompt "шаг по времени = ": dt
   LET dt2 = dt*dt
   LET nsnap = 5              ! число шагов по времени между снимками
   INPUT prompt "максимальная начальная скорость = " : vmax
   LET pos_row = Ly/3
   LET pos_col = Lx/8                      ! конфигурация для рис. 6.4
```

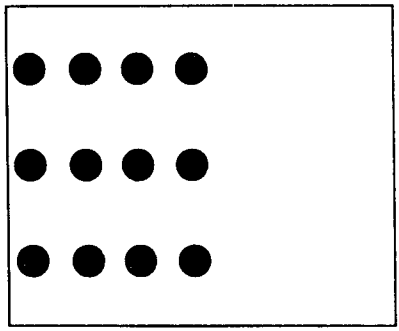

Рис. 6.4. Начальные условия, использованные в задаче 6.1 с параметрами, приведенными в подпрограмме initial.

```
    FOR row = 1 to 3
       FOR col = 1 to 4
          LET i = i + 1
          LET x(i) = pos_col*(col - 0.5)
          LET y(i) = pos_row*(row - 0.5)
          ! choose random  velocities
          LET vx(i) = vmax*(2*rnd - 1)
          LET vy(i) = vmax*(2*rnd - 1)
       NEXT col
    NEXT row
    FOR i = 1 to N
      LET vxcum = vxcum + vx(i) ! полная скорость (импульс) центра масс
      LET vycum = vycum + vy(i)
    NEXT i
    LET vxcum = vxcum/N
    LET vycum = vycum/N
    FOR i = 1 to N
       LET vx(i) = vx(i) - vxcum
       LET vy(i) = vy(i) - vycum
    NEXT i
END SUB
```

В подпрограмме screen мы описываем параметры, необходимые для вывода на экран всех N частиц. Частицы изображаются в виде сплошных

кружков, а не «точек». Сплошные кружки запоминаются в переменной *ball$*.

```
SUB screen(x(),y(),N,Lx,Ly,ball$,r)
   LET mx = 0.1*Lx                    ! предельные размеры
   LET my = 0.1*Ly
   LET aspect_ratio = 1.5
   LET mx = aspect_ratio*mx
   LET Sx = aspect_ratio*Lx
   LET Sy = Ly
   SET window -mx,Sx+mx,-my,Sy+my
   BOX LINES 0,Lx,0,Ly
   LET r = 0.2                        ! рисуем кружок, изображающий частицу
   BOX CIRCLE x(1)-r,x(1)+r,y(1)-r,y(1)+r
   FLOOD x(1),y(1)
   BOX KEEP x(1)-r,x(1)+r,y(1)-r,y(1)+r in ball$
   FOR i = 1 to N
      BOX SHOW ball$ at x(i)-r,y(i)-r
   NEXT i
END SUB
```

Затем мы реализуем алгоритм Верле для численного решения уравнений движения Ньютона. Обратите внимание на то, что скорость частично обновляется с использованием старого ускорения. Далее вызывается подпрограмма **accel**, которая вычисляет ускорение, используя новую координату, и скорость еще раз изменяется. Подпрограмма **Verlet** обращается к подпрограмме **periodic**, которая в свою очередь обеспечивает рассмотрение частиц только в центральной ячейке.

```
SUB Verlet(x(),y(),vx(),vy(),ax(),ay(),N,Lx,Ly,dt,dt2,virial,xflux,yflux,pe,ke)
   FOR i = 1 to N
      LET xnew = x(i) + vx(i)*dt + 0.5*ax(i)*dt2
      LET ynew = y(i) + vy(i)*dt + 0.5*ay(i)*dt2
      ! при необходимости возвращаем частицу в центральную ячейку
      CALL periodic(xnew,ynew,Lx,Ly)
      LET x(i) = xnew
      LET y(i) = ynew
   NEXT i
   FOR i = 1 to N
      ! частично изменяем скорость, используя старое ускорение
```

```
      LET vx(i) = vx(i) + 0.5*ax(i)*dt
      LET vy(i) = vy(i) + 0.5*ay(i)*dt
   NEXT i
   CALL accel(x, y, ax, ay, N, Lx, Ly, pe)          ! вычисляем новое ускорение
   FOR i = 1 to N
      ! частично изменяем скорость, используя новое ускорение
      LET vx(i) = vx(i) + 0.5*ax(i)*dt
      LET vy(i) = vy(i) + 0.5*ay(i)*dt
      LET ke = ke + .5*(vx(i)*vx(i) + vy(i)*vy(i))
   NEXT i
END SUB
```

В подпрограмме **accel** для нахождения полной силы, действующей на каждую частицу, используется третий закон Ньютона. (Напомним, что в нашей системе единиц масса частицы равна единице.) Обращение к подпрограмме **separation** обеспечивает, что расстояние между частицами никогда не будет больше $Lx/2$ в x-направлении и $Ly/2$ в y-направлении.

```
SUB accel(x(), y(), ax(), ay(), N, Lx, Ly, pe)
  FOR i = 1 to N
     LET ax(i) = 0
     LET ay(i) = 0
  NEXT i
  FOR i = 1 to (N - 1) ! вычисляем полную силу, действующую на частицу i
     FOR j = (i + 1) to N
        LET dx = x(i) - x(j)
        LET dy = y(i) - y(j)
        CALL separation(dx, dy, Lx, Ly)
        LET r = sqr(dx*dx + dy*dy)
        CALL f(r, force, potential)
        LET ax(i) = ax(i) + force*dx
        LET ay(i) = ay(i) + force*dy
        LET ax(j) = ax(j) - force*dx                ! третий закон Ньютона
        LET ay(j) = ay(j) - force*dy
        LET pe = pe + potential
     NEXT j
  NEXT i
END SUB
```

```
SUB separation(dx, dy, Lx, Ly)
   ! функция sgn выдает знак своего аргумента
   IF abs(dx) > .5*Lx then LET dx = dx - sgn(dx)*Lx
   IF abs(dy) > .5*Ly then LET dy = dy - sgn(dy)*Ly
END SUB

SUB periodic(xtemp, ytemp, Lx, Ly)
   IF xtemp < 0 then LET xtemp = xtemp + Lx
   IF xtemp > Lx then LET xtemp = xtemp - Lx
   IF ytemp < 0 then LET ytemp = ytemp + Ly
   IF ytemp > Ly then LET ytemp = ytemp - Ly
END SUB
```

Потенциал и сила вычисляются в подпрограмме f.

```
SUB f(r, force, potential)
   LET ri = 1/r
   LET ri3 = ri*ri*ri
   LET ri6 = ri3*ri3
   LET g = 24*ri*ri6*(2*ri6 - 1)
   LET force = g/r     ! множитель 1/r компенсируется dx, dy для Fx, Fy
   LET potential = 4*ri6*(ri6 - 1)
END SUB
```

Координаты частиц выдаются на экран в подпрограмме **snapshot**. Частота, с которой координаты частиц должны обновляться на экране, зависит от Δt.

```
SUB snapshot(x(), y(), N, Lx, Ly, ball$, r)
  CLEAR
  FOR i = 1 to N
    BOX SHOW ball$ at x(i) - r, y(i) - r
  NEXT i
  BOX LINES 0, Lx, 0, Ly
END SUB
```

ЗАДАЧА 6.1. Приближение к равновесию разреженного газа

а. Рассмотрите $N = 12$ частиц, взаимодействующих в ящике с линейными размерами $Lx = Ly = 8$ с потенциалом взаимодействия Леннар-

да—Джонса. При данном выборе N и L плотность равна $\rho^* = 12/64 \approx$ ≈ 0.19. Предположим, что в начальный момент времени частицы удерживаются в левой половине ящика и расположены в узлах прямоугольной сетки, как показано на рис. 6.4. В момент $t = 0$ связь удаляется и частицы начинают перемещаться свободно по всему ящику. Возьмите максимальную начальную скорость частиц $v_{max} = 1.0$ и $\Delta t = 0.02$. Оптимальный выбор таких параметров, как *nsnap* (число временных шагов между «снимками» частиц), зависит от вычислительной производительности вашего компьютера, а также быстродействия и разрешения вашего графического дисплея. Учтите, что на большинстве персональных компьютеров вывод графических изображений занимает относительно много времени. Просмотрите достаточное число снимков, чтобы частицы заметно передвинулись из своих начальных положений. (Это может занять порядка 100—200 шагов по времени.) Стала ли система более или менее хаотичной?

б. По наблюдаемым снимкам траекторий частиц оцените время, за которое система достигает равновесия. Что служит для вас качественным критерием для «равновесия»?

в. Вычислите $n(t)$—число частиц в левой половине ящика. Нарисуйте $n(t)$ как функцию времени. Как качественно ведет себя $n(t)$? Сколько частиц в среднем находится слева?

***г.** Запомните координаты и скорости всех частиц в момент времени t, когда $n(t) \sim 6$. (Выведите на печать координаты и скорости $vx(i)$ и $vy(i)$ или же воспользуйтесь подпрограммой **save_config**, приведенной перед задачей 6.4.) Затем рассмотрите обратный по времени процесс, т.е. движение, которое имело бы место, если бы направление времени было обратным. Этот процесс эквивалентен заданию $\mathbf{v} \longrightarrow -\mathbf{v}$ для всех частиц. Используйте программу **md** с начальными условиями $vx(i) \longrightarrow -vx(i)$ и $vy(i) \longrightarrow -vy(i)$ и опишите качественно зависимость $n(t)$. Каково состояние системы при $t = 0$?

ЗАДАЧА 6.2. Специальные начальные условия

а. Рассмотрите начальные условия, соответствующие $N = 11$, $Lx = Ly = 10$, $x(i) = Lx/2$, $y(i) = (i - 0.5)*Ly/N$, $vx(i) = 1$ и $vy(i) = 0$ (рис. 6.5). Исключите из подпрограммы **initial** инструкции, которые обеспечивают равенство нулю полного импульса. Дос-

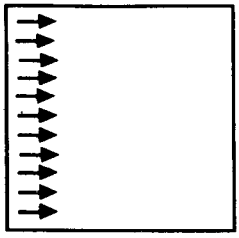

Рис. 6.5. Пример «специального» начального условия; стрелки показывают величину и направление скорости каждой частицы.

тигает ли система в конце концов равновесия или нет? Почему?

б. Поменяйте направление скорости частицы 4 так, чтобы $vx(4) = 0.5$ и $vy(4) = sqr(1 - vx(4)*vx(4))$. Достигает ли система в конце концов равновесия? Объясните, почему «почти» все начальные состояния приводят к качественно одинаковому поведению.

На снимках, формируемых в задаче 6.1, система видна с мельчайшими возможными деталями. С такой *микроскопической* точки зрения эти снимки кажутся довольно сложными, поскольку координаты частиц на каждом снимке разные. Однако ясно, что если мы не интересуемся траекториями каждой отдельной частицы, то систему можно описать проще. А именно, макроскопическое состояние, или *макросостояние*, системы можно в какой-то степени характеризовать числом частиц в любой части ящика. Ваши наблюдения временно́го поведения числа частиц в левой половине ящика не должны противоречить двум результатам, носящим общий характер:

1. После удаления внутренней перегородки изолированная система изменяется во времени от «менее случайного» состояния к «более случайному».
2. Конечное макроскопическое состояние характеризуется относительно малыми флуктуациями около среднего, которые не зависят от времени.

Говорят, что система многих частиц находится в *равновесии*, если ее макроскопическое состояние не зависит от времени. Процесс называют *необратимым*, если обратный по времени процесс ($t \longrightarrow -t$) чрез-

вычайно маловероятен. Например, если мы опрокинем чашку кофе (ни в коем случае не рядом с компьютером!), мы знаем, что кофе будет вытекать из чашки. В принципе возможен обратный по времени процесс, но он чрезвычайно маловероятен. Обратите внимание на то, что в законах движения не содержится ничего, что указывало бы на выделенное направление времени (см. задачу 6.1г).

Приближение к равновесию мы охарактеризовали качественно, а именно, что постоянное движение частиц почти всегда дает в результате однородное распределение, которое заполняет весь ящик. Одной из макроскопических характеристик «случайности» системы многих частиц является энтропия, о которой пойдет речь в гл. 14.

Прежде чем двигаться дальше, имело бы смысл посмотреть, как ведет себя полная энергия, и проверить наше утверждение, что алгоритм Верле обеспечивает сохранение энергии при разумном выборе Δt.

ЗАДАЧА 6.3. Алгоритм Верле и сохранение энергии

Используйте программу **md** с $N = 12$, $vmax = 0.2$, $Lx = Ly = 8$. Расположите частицы в виде прямоугольной сетки, покрывающей ящик, и выберите скорости случайным образом. Определите значение Δt, необходимое для сохранения полной энергии с точностью до 5% на протяжении 200 шагов по времени. (В большинстве научных приложений энергия сохраняется с точностью 1% или лучше.) Сохраняются ли по отдельности кинетическая и потенциальная энергия? Указания: для вычисления кинетической энергии добавьте в соответствующее место подпрограммы **Verlet** следующую инструкцию:

```
LET ke = ke + 0.5*(vx(i)*vx(i) + vy(i)*vy(i))
```

Для вычисления потенциальной энергии добавьте в подпрограмму **accel** следующую инструкцию:

```
LET pe = pe + potential
```

Подпрограмма **output** вызывается из главной программы и вычисляет среднюю кинетическую и потенциальную энергию после каждого снимка.

```
SUB output(N, Lx, Ly, virial, xflux, yflux, pe, ke, nsnap, itime)
  ! накопленные значения ke и pe вычислялись nsnap раз
  LET ke = ke/nsnap
  LET pe = pe/nsnap
  LET E = (ke + pe)/N                    ! полная энергия на частицу
  SET cursor 1,1
  PRINT "энергия на частицу = "; E
  SET cursor 1,50
  PRINT "ke = "; ke
  SET cursor 2,50
  PRINT "pe = "; pe
  SET CURSOR 4,50
  PRINT "время = "; itime
  LET ke = 0                             ! начальная установка переменных
  LET pe = 0
END SUB
```

6.6. ИЗМЕРЕНИЕ МАКРОСКОПИЧЕСКИХ ВЕЛИЧИН

Макросостояние газа мы характеризовали числом частиц в левой половине ящика. Известно, что равновесное макросостояние можно характеризовать и другими параметрами, такими как абсолютная температура T, среднее давление P, а также объемом и полной энергией.

Кинетическое определение температуры вытекает из теоремы о равнораспределении: каждый квадратичный член, входящий в выражение для энергии классической системы, находящейся в равновесии при температуре T, имеет среднее значение $\frac{1}{2}k_BT$, где k_B — постоянная Больцмана. Отсюда мы можем определить температуру T системы в d-мерном пространстве соотношением

$$\frac{d}{2}Nk_BT = \sum \frac{1}{2}\langle mv_i^2\rangle, \tag{6.4}$$

где сумма берется по всем N частицам системы и d компонентам скорости. Скобки $\langle\ldots\rangle$ обозначают усреднение по времени. Выражение (6.4) представляет собой пример связи макроскопической величины, в данном случае температуры, с временным средним по траекториям частиц. Заметим, что соотношение (6.4) справедливо только в том случае, если движение центра масс системы равно нулю — не хватает еще, чтобы движение центра масс резервуара изменяло температуру! В системе СИ

температура T измеряется в градусах Кельвина (К) и постоянная $k_B = 1.38 \cdot 10^{-23}$ Дж/К. В дальнейшем мы будем измерять температуру в единицах ε/k_B.

Еще одной тепловой характеристикой является *теплоемкость* при постоянном объеме $C_V = (\partial E/\partial T)_V$, где E — полная энергия. C_V есть мера количества тепла, необходимого, чтобы произвести изменение температуры. Поскольку теплоемкость зависит от размеров системы, удобно определить *удельную теплоемкость* на частицу, а именно $c_V = C_V/N$. Легче всего получить c_V путем нахождения средней потенциальной энергии при соседних температурах T и $T + \Delta T$; c_V складывается из температурной зависимости потенциальной энергии и удельной теплоемкости, связанной с кинетической энергией, т.е. $(d/2)k_B$.

Чтобы определить среднее давление, предположим что частицы находятся в резервуаре с жесткими стенками. Как известно, столкновения частиц со стенками резервуара приводят к тому, что на каждый элемент стенки действует средняя результирующая сила. Средняя сила, действующая на единичную площадку, и есть давление P газа. Давление можно найти, связывая силу в горизонтальном направлении со скоростью изменения соответствующей компоненты импульса частиц, ударяющихся о стенку.

Из тех же соображений можно получить среднее давление в отсутствие стенок. Поскольку давление в равновесном состоянии во всех направлениях одинаково, мы можем связать давление с общим переносом количества движения через элемент поверхности в любом месте системы. Рассмотрим элемент поверхности dA и предположим, что среднее количество движения, пересекающее в единицу времени нашу поверхность слева направо, равно $\mathbf{S}_+$, а $\mathbf{S}_-$ — среднее количество движения, пересекающее поверхность в единицу времени справа налево. Тогда средняя сила $\mathbf{F}$ равна

$$\mathbf{F} = \mathbf{S}_+ - \mathbf{S}_- \tag{6.5}$$

и среднее давление определяется выражением

$$P = \frac{dF_n}{dA}, \tag{6.6}$$

где F_n обозначает компоненту силы, нормальную к элементу поверхности. (В двумерном случае давление равно плотности потока импульса через единичный отрезок, а не через единичную площадку.) Что служит

соответствующей мерой давления для потенциала Леннарда—Джонса?

Один из способов реализации этого метода определения среднего давления состоит в следующем. Представим себе, что на «ребрах» элементарной ячейки установлены четыре воображаемые поверхности. Тогда мы можем модифицировать подпрограмму **periodic** и вычислять поток импульса за один шаг по времени.

```
SUB periodic(xnew, ynew, px, py, Lx, Ly, xflux, yflux)
  ! модификация подпрограммы для нахождения потока
  ! задайте px = vx(i) и py = vy(i) в подпрограмме Verlet
  IF xnew < 0 then
      LET xnew = xnew + Lx
      LET xflux = xflux - px              ! перенос количества движения
  END IF
  IF xnew > Lx then
      LET xnew = xnew - Lx
      LET xflux = xflux + px
  END IF
  IF ynew < 0 then
      LET ynew = ynew + Ly
      LET yflux = yflux - py
  END IF
  IF ynew > Ly then
      LET ynew = ynew - Ly
      LET yflux = yflux + py
  END IF
END SUB
```

Среднее давление можно найти, добавив в подпрограмму **output** следующие инструкции:

```
LET presure = ( (xflux/(2*Lx)) + (yflux/(2*Ly)) )/(dt*nsnap) ! метод потока
LET xflux = 0                  ! задаем начальные значения переменных и
LET yflux = 0                  ! передаем их в головную программу
```

Другой способ вычисления давления вытекает из теоремы вириала:

$$PV = Nk_BT + \frac{1}{d} <\sum_{i}^{N} \mathbf{r}_i \cdot \mathbf{F}_i>, \qquad (6.7)$$

где $\mathbf{r}_i$ — координата i-й частицы, $\mathbf{F}_i$ — полная сила, действующая на частицу i со стороны всех остальных частиц, и сумма берется по всем N частицам. Вывод формулы (6.7) дан в приложении 6А. Чтобы с помощью вириала вычислить давление, добавим в подпрограмму **Verlet** следующую инструкцию:

```
LET virial = virial + x(i)*ax(i) + y(i)*ay(i)
```

Для вычисления среднего давления добавим в подпрограмму **output** следующие инструкции:

```
LET pvirial = (N*T)/(Lx*Ly) + 0.5*virial/(Lx*Ly*nsnap)
LET virial = 0
```

Итак, мы встретили уже два примера — (6.4) и (6.7), содержащие связь макроскопических величин со средними по времени от функций координат и скоростей частиц системы. В равновесном состоянии эти средние не зависят от времени. В гл. 15 и 16 мы будем рассматривать второй вид усреднения — среднее по статистическим ансамблям. Связь этих двух методов усреднения кратко обсуждается в гл. 15.

При моделировании методом молекулярной динамики процесс установления равновесия зачастую может занимать существенную часть от общего времени счета. Как правило, в качестве начальных условий удобнее всего выбирать «равновесную» конфигурацию из какого-либо прежнего расчета, которая отвечает температуре и плотности, близким к требуемым. Следующая подпрограмма запоминает последнюю конфигурацию пропускаемого варианта и может быть использована в качестве последней подпрограммы, вызываемой программой **md**.

```
SUB save_config(x(),y(),vx(),vy(),N,Lx,Ly)
  INPUT prompt "имя файла под последнюю конфигурацию = ": file$
  OPEN #1: name file$, access output, create new
  PRINT #1: N, ",", Lx, ",", Ly
  FOR i = 1 to N
     PRINT #1: x(i), ",", y(i)
     PRINT #1: vx(i), ",", vy(i)
  NEXT i
  CLOSE #1
END SUB
```

Следующие инструкции можно добавить в подпрограмму **initial**, с тем чтобы можно было воспользоваться какой-то прежней равновесной конфигурацией в качестве начальной конфигурации нового расчета.

```
INPUT prompt "новая конфигурация (да/нет – y/n)? ": new$
IF new$ = "n" or new$ = "N" then
   INPUT prompt "имя файла? ": file$
   OPEN #1: name file$, access input
   FOR i = 1 to N
      INPUT #1: x(i),y(i)
      INPUT #1: vx(i),vy(i)
   NEXT i
   CLOSE #1
ELSE
! выбираем начальные условия, как раньше
END IF
```

Необходимо только один раз провести моделирование молекулярной динамики, отправляясь от системы с произвольной конфигурацией. Последующие расчеты для различных состояний можно начинать, модифицируя условия, сформированные на момент завершения соответствующего предыдущего расчета. Вопрос о том, как изменять температуру системы при неизменной плотности, рассмотрен в задаче 6.6.

ЗАДАЧА 6.4. Качественные свойства жидкости и газа

а. Во всех пунктах данной задачи для получения своей начальной конфигурации используйте следующие инструкции data и read. Инструкция **READ** читает элементы данных из инструкций **DATA** и присваивает их значения соответствующим переменным. Каждая инструкция **READ** читает очередной элемент из инструкции **DATA**. Напишите короткую программу, в которой используются инструкции **READ** и **DATA**, и удостоверьтесь, что вы правильно понимаете работу этих инструкций.

```
DATA 0.4,0.8,3.1,0.9,5.2,1.3,5.4,5.9,0.4,2.3,1.6,2.1,3.9,1.6,5.3,2.3
DATA 5.4,4.5,2.3,4.1,3.7,4.0,4.9,3.3,0.8,4.4,2.8,5.3,4.3,0.6,4.4,4.6
  FOR i = 1 to N
    READ x(i),y(i)
    LET x(i) = rscale*x(i)
    LET y(i) = rscale*y(i)
    LET vx(i) = 0
    LET vy(i) = 0
  NEXT i
```

Выберите $N = 16$, $Lx = Ly = 6$, *rscale* = 1 и $\Delta t = 0.02$. Чему равна приведенная плотность? Выберите *nsnap* = 20 и проведите моделирование на протяжении по крайней мере 200 шагов по времени. Чему равна начальная температура системы? Каков характер ваших снимков? Как долго система достигает равновесия? Каков ваш критерий равновесия? Вычислите полную энергию системы, температуру и давление. Не учитывайте при усреднении по времени неравновесные конфигурации.

б. Как было найдено в п. «а», полная энергия отрицательна. Какой смысл имеет знак полной энергии? Повторите моделирование с теми же начальными условиями (*rscale* = 1), но с $Lx = Ly = 30$. Чему равна полная энергия в этом случае? Тому же, что и в п. «а»? Если нет, то почему? Опишите характер полученных снимков. Заполняют ли частицы ящик равномерно за время порядка 200 шагов или у них наблюдается тенденция к образованию «капелек»?

в. Увеличьте начальное расстояние между частицами. Используйте $Lx = Ly = 30$ и *rscale* = 2. Чему равна приведенная плотность системы? Оцените начальную плотность капельки. Какая получилась полная энергия — отрицательная или положительная? Если величину *rscale* сделать достаточно большой, капелька должна в конце концов «испариться», даже если полная энергия отрицательна. В реальной системе капелька находилась бы в равновесии со своим паром и обе фазы существовали бы совместно. По мере уменьшения плотности начальной капельки все больше частиц внутри капельки будет испаряться и размер капельки будет сокращаться. Разумеется, нужно с осторожностью относиться к любым заключениям о структуре системы, основанным на моделировании только с 16 частицами.

г. Положите $Lx = Ly = 30$ и $rscale = 0.8$. Чему равна приведенная плотность системы? Оцените начальную плотность капельки. Поскольку вначале частицы ближе друг к другу, необходимо выбрать меньшее значение Δt, например $\Delta t = 0.01$. Почему полная энергия положительна? Пройдите по крайней мере 200 шагов по времени и вычислите средние температуру и давление по последним 100 шагам. Запомните конечную конфигурацию, с тем чтобы использовать ее в задаче 6.5. Каков характер ваших снимков? Объясните, почему данное равновесное состояние можно трактовать как газ.

ЗАДАЧА 6.5. Распределение по скоростям

а. В качестве начальной конфигурации используйте для этой задачи конечную конфигурацию из задачи 6.4г. Наша цель — вычислить для равновесного состояния вероятность $P(v)\Delta v$ того, что скорость частицы заключена между v и $v + \Delta v$. Исходя из начальной конфигурации, оцените максимальную скорость частиц. Выберите интервалы шириной Δv, при этом номер интервала k равен $v/\Delta v$, где v — скорость частицы. Целесообразно выбрать Δv равным $0.1 * sqr(T)$, где T — температура. Для регистрации количества попаданий скоростей частиц в соответствующий интервал с номером k используйте массив $prob(k)$:

```
LET prob(k) = prob(k) + 1
```

Предположим, что $\Delta v = 0.1$, и рассмотрим скорости $v = 0.3$, 0.49, 0.5, 0.51 и 0.9. Каким значениям k они соответствуют? Напишите короткую программу для получения массива $prob(k)$. Определите $prob(k)$ после каждого временно́го шага и вычислите средние значения $prob(k)$ по крайней мере по 100 шагам по времени. Нормируйте $prob(k)$, поделив на число частиц и количество шагов по времени. Заметим, что наблюдать траектории частиц нет необходимости. Напечатайте таблицу, содержащую значения k, v и $prob(k)$.

б. Постройте график плотности вероятности $P(v)$ как функции от v. Каков качественный вид $P(v)$? Чему равно наиболее вероятное значение v? Какова приблизительно «ширина» $P(v)$? Данное распределение вероятности называется распределением Максвелла — Больцмана.

в. Определите распределение вероятности для каждой компоненты скорости. Удостоверьтесь, что вы различаете положительные и от-

рицательные скорости. Каковы наиболее вероятные значения для x- и y-компонент? Чему равны средние значения?

ЗАДАЧА 6.6. Температурная зависимость внутренней энергии

а. Одной из характерных черт метода молекулярной динамики является то, что полная энергия определяется начальными условиями, а температура есть величина производная, определяемая только после достижения системой теплового равновесия. В результате трудно изучать систему, находящуюся при конкретной температуре. Обычно для достижения требуемой температуры T_f в качестве начального условия берут равновесную конфигурацию при температуре T_i, по возможности наименее отличающуюся от T_f. Определяют «масштабный» множитель f из соотношения $T_f = fT_i$ и пересчитывают скорости по формуле $\mathbf{v} \rightarrow f^{1/d}\mathbf{v}$. Для достижения T_f может требоваться и не одно перемасштабирование скоростей. В качестве начальной конфигурации системы возьмите равновесную конфигурацию из задачи 6.5а с параметрами $Lx = Ly = 6$, $N = 16$, $rscale = 1$ и $\Delta t = 0.01$. Задайте $f = 1.2$ и найдите полную энергию и новую равновесную температуру. Предусмотрите для выхода на равновесие по крайней мере 100 шагов по времени. После того как равновесие установилось, усредните кинетическую энергию на частицу по 200 временны́м шагам, чтобы получить приемлемую оценку средней температуры системы. Вычислите также временну́ю зависимость равновесной температуры путем усреднения кинетической энергии на частицу по пяти шагам по времени. Повторите это перемасштабирование еще четыре раза и вычислите для каждого случая полную энергию, среднюю температуру и равновесные флуктуации температуры.

б. Сравните начальную и конечную температуры в п. «а». Связаны ли они приближенным соотношением $T_f = fT_i$?

в. По своим данным $T(E)$, найденным в п. «а», начертите график зависимости полной энергии E от T. Оцените вклад в c_V — удельную теплоемкость — от потенциальной энергии и от кинетической. Какая часть удельной теплоемкости обусловлена потенциальной энергией? Почему трудно добиться точных вычислений c_V?

г. Изобразите равновесную температуру, усредненную по пяти временны́м шагам, как функцию времени для каждой энергии, рассмот-

ренной в п. «а». Почему равновесная температура флуктуирует? Для температур, рассмотренных в п. «а», оцените визуально величину флуктуаций температуры. Как качественно ведут себя флуктуации температуры в зависимости от T?

В задаче 6.7 мы вычислим давление и, следовательно, *уравнение состояния* (соотношение между давлением, температурой и объемом) газа.

ЗАДАЧА 6.7. Уравнение состояния неидеального газа

а. Выберите в качестве начальной конфигурации равновесную конфигурацию из задачи 6.6а ($Lx = Ly = 6$, $N = 16$, $rscale = 1$, $\Delta t = 0.01$). С помощью программы **md** и подпрограммы **output** вычислите среднее давление, используя метод суммарного потока импульса и теорему вириала. Предусмотрите по крайней мере 100 шагов по времени для установления равновесия и 200 шагов для расчета средних. Получается ли давление постоянным или оно флуктуирует? Как согласуется это давление с результатом для идеального газа? Какой метод вычисления давления точнее? Свой ответ обоснуйте.

*б. Плотные газы и жидкости описываются приближенно уравнением состояния *Ван-дер-Ваальса*:

$$P = k_B T \frac{\rho}{1 - b\rho} - a\rho^2. \qquad (6.8)$$

Феноменологические параметры b и a связаны с отталкивающей и притягивающей частями взаимодействия соответственно и приближенно не зависят от температуры. Используйте те же конфигурации, что в задаче 6.6а, и найдите зависимость P от T. Представьте график P от T и, используя формулу (6.8), получите оценку для параметров a и b.

*в. Измените плотность, как это делалось в задаче 6.4а, используя $rscale = 1.2$. Вычислите давление для различных значений T и оцените a и b. Сильно ли меняются значения a и b? Оцените грубо погрешности своих вычислений.

Несмотря на то что использование периодических краевых условий

минимизирует поверхностные эффекты, важно еще так выбрать симметрию центральной «элементарной ячейки», чтобы она соответствовала симметрии твердой фазы системы. Этот выбор ячейки существен, если мы желаем максимально реалистично промоделировать свойства системы высокой плотности, низкой температуры. Если ячейка не соответствует истинной кристаллической структуре, частицы не смогут образовать идеальный кристалл и некоторые из частиц будут блуждать в нескончаемых попытках найти свои «правильные» позиции. Следовательно, численное моделирование небольшого числа частиц при высокой плотности и низкой температуре привело бы к ложным результатам.

Мы знаем, что равновесная структура кристаллического твердого тела при $T = 0$ представляет собой конфигурацию с наименьшей энергией. В задаче 6.8 мы подтверждаем, что состоянию с наименьшей энергией для двумерного твердого тела Леннарда—Джонса отвечает треугольная решетка, а не квадратная (рис. 6.6).

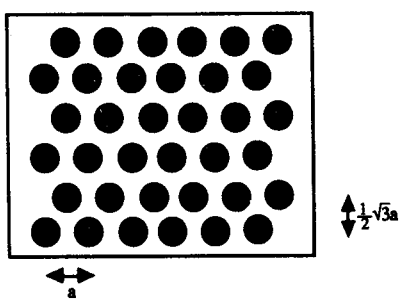

Рис. 6.6. В двумерной системе Леннарда—Джонса состоянию с наименьшей энергией отвечает треугольная, а не квадратная решетка (см. задачу 6.8).

*ЗАДАЧА 6.8. Энергия основного состояния двумерных решеток

а. Общий вид треугольной решетки можно понять из рис. 6.6. Каждый узел имеет шесть ближайших соседей. Напишите программу, которая сравнивает энергию, приходящуюся на частицу системы N частиц, взаимодействующих с потенциалом Леннарда—Джонса. Рассмотрите как треугольную, так и квадратную решетки. Пусть ширина треугольной решетки равна L_x, а n_c—число частиц в каждом столбце или строке. Столбцы треугольной решетки отстоят друг от друга

на расстоянии $a = L_x/n_c$, а каждую строку разделяет расстояние $\sqrt{3}a/2$. В каждой строке узлы смещены на $a/2$ относительно предыдущей строки. Высота треугольной решетки составляет $L_y = \frac{\sqrt{3}}{2} L_x$ и $N = n_c^2$. Чтобы обе решетки имели одинаковую плотность, возьмите линейный размер квадратной решетки равным $L = (L_x L_y)^{1/2}$. Хотя в качестве элементарной ячейки треугольной решетки можно выбрать ромб, удобнее выбирать элементарную ячейку прямоугольной. (Для треугольной решетки не существует ни одной квадратной элементарной ячейки.) Возьмите $n_c = 6$ и определите энергию каждой решетки для $L_x = 5$ и 7. Чему равна плотность системы в каждом случае? Зависят ли ваши результаты для E/N от размера решетки? Энергия какой решетки меньше? Объясните свои результаты с точки зрения более плотной упаковки частиц в треугольной решетке.

б. Рассмотрите $n_c = 6$ и $L_x = 10$, 20 и $2^{1/6} n_c$. Решетки какой структуры имеют наименьшую энергию в каждом случае? Во всех этих вариантах суммарная сила, действующая на частицу, равна нулю. Почему? Последняя плотность соответствует случаю, когда сила между ближайшими соседями равна нулю. Убедитесь, что при этой плотности решетка неустойчива, поскольку сила, действующая на любую частицу, отклонившуюся от своего узла решетки, является притягивающей. Следовательно, система будет коллапсировать к большей плотности, где треугольная решетка всегда предпочтительнее.

в. Повторите предыдущие вычисления для $V(r) = 1/r^6$ и $1/r^4$. По-прежнему ли треугольная решетка предпочтительнее?

ЗАДАЧА 6.9. Твердое состояние и плавление

а. Задайте параметры $N = 16$, $\Delta t = 0.01$, $L_x = 4$ и $L_y = 2\sqrt{3}$. Поместите частицы в узлы треугольной решетки, покрывающей весь резервуар, и сообщите каждой частице нулевую начальную скорость. Измерьте температуру и давление как функцию времени и сохраните несколько «снимков» системы. Чему равна полная энергия системы? Остается ли система в твердом состоянии?

б. Сообщите частицам случайные скорости из интервала $[-0.1, 0.1]$. Чему равна полная энергия? Подождите установления в системе рав-

новесия примерно на протяжении 50 шагов по времени и усредните температуру и давление по 100 временны́м шагам. Получите снимки системы с интервалом 25 шагов по времени. Запомните равновесную конфигурацию для использования ее в п. «в».

в. Выберите в качестве начальной конфигурации равновесную конфигурацию из п. «б», но увеличьте температуру, положив $v_i \longrightarrow 2v_i$ Чему равна новая полная энергия? Получите снимки системы с интервалом 25 шагов по времени и опишите качественные особенности движения частиц в системе. Чему равны равновесная температура и давление системы? После установления равновесия увеличьте опять температуру, пересчитав скорости таким же способом, как выше. Повторите этот пересчет и измерьте $P(T)$ и $E(T)$ для шести разных температур.

г. По результатам, полученным в п. «в», постройте графики $E(T) - E(0)$ и давления как функции T. Пропорциональна ли зависимость $E(T) - E(0)$ температуре T? Чему равна средняя потенциальная энергия для гармонического твердого тела? Согласуются ли результаты ваших измерений температурной зависимости $E(T) - E(0)$ с тем, что вы ждали?

д. Уменьшите плотность, умножив L_x, L_y и координаты частиц на 1.1. Какие у вас получились температура и давление? Каков характер снимков? Повторяйте перемасштабирование плотности и координат до тех пор, пока система не расплавится. По каким качественным признакам вы распознаете на своих снимках «плавление»?

6.7. ПРОСТЫЕ СВОЙСТВА ПЕРЕНОСА

Теперь мы обсудим некоторые динамические свойства жидкости, находящейся в равновесном состоянии. Предположим, что мы следим за траекторией какой-то одной *пробной* частицы (частица i) и что в некий произвольно выбранный момент времени t_1 ее координата равна $\mathbf{r}_i(t_1)$. В некоторый последующий момент времени t_2 мы можем определить ее смещение $\mathbf{r}_i(t_2) - \mathbf{r}_i(t_1)$. Известно, что если суммарная сила, действующая на частицу i, равна нулю, то ее смещение растет со временем линейно. Однако в жидкости каждая частица претерпевает множество соударений, и в среднем ее суммарное смещение будет равно нулю. Го-

раздо больший интерес представляет величина среднего квадрата смещения $R(t)^2$, определяемая формулой

$$R(t)^2 = \langle |\mathbf{r}_i(t_2) - \mathbf{r}_i(t_1)|^2 \rangle, \tag{6.9}$$

где $t = t_2 - t_1$, и усреднение проведено по всем частицам. Поскольку система находится в равновесии, начало отсчета времени является произвольным и среднее в (6.9) зависит только от разности времен t. В задаче 6.10 мы убедимся, что зависимость от t функции $R(t)$ удовлетворяет формуле Эйнштейна

$$R(r)^2 = 2dDt \qquad (t \longrightarrow \infty), \tag{6.10}$$

где D — *коэффициент самодиффузии*. Заметим, что для достаточно больших t величина $R(t)$ растет, как $t^{1/2}$.

Другой характеристикой одной частицы является «автокорреляционная функция скорости» $Z(t)$, определение которой мы сейчас поясним. Предположим, что частица i в момент времени t_1 имеет скорость $\mathbf{v}_i(t_1)$. Если суммарная сила, действующая на частицу i, равна нулю, то скорость последней будет оставаться постоянной, и, следовательно, скорость в любой последующей момент времени будет «связана» — коррелирована со скоростью в начальный момент. Однако за счет взаимодействия с другими частицами жидкости скорость частицы будет изменяться, и можно ожидать, что после некоторого числа столкновений ее скорость не будет сильно коррелирована со своей начальной скоростью. В результате определение $Z(t)$ имеет вид

$$Z(t) = \langle \mathbf{v}_i(t_2) \cdot \mathbf{v}_i(t_1) \rangle, \tag{6.11}$$

где $t = t_2 - t_1$. Если определять коэффициент самодиффузии D формулой (6.9), то можно показать, что D и $Z(t)$ связаны соотношением

$$D = \frac{1}{d} \int_{-\infty}^{+\infty} Z(t)\, dt. \tag{6.12}$$

Соотношение (6.12), связывающее коэффициент самодиффузии с интегралом по времени от автокорреляционной функции, есть пример общей связи между коэффициентами переноса, такими как вязкость и теплопроводность, и автокорреляционными функциями.

Автокорреляционная функция скорости и средний квадрат смещения

вычисляются в подпрограмме **transport**. Эту подпрограмму следует вызывать тогда же, когда вычисляются давление и температура. Для хранения координат и скоростей частиц используются четыре массива: *xs*, *ys*, *vxs* и *vys*. Переменная *ntrans* содержит количество обращений к подпрограмме **transport** и используется для нормировки результатов в подпрограмме **trans_output**. Последняя подпрограмма вызывается в самом конце головной программы.

```
SUB transport(N,Lx,Ly,x(),y(),vx(),vy(),xs(,),ys(,),vxs(,),vys(,),ntrans,Z(),R2())
  ! расчет функции автокорреляции скорости и среднего квадрата смещения
  ! перед вычислением Z(t) и R2(t) запоминается 10 конфигураций
  IF ntrans > 10 then          ! вычисляем функции переноса
     FOR itime = 1 to 10
        FOR i = 1 to N
           LET dx = x(i) - xs(i,itime)
           LET dy = y(i) - ys(i,itime)
           CALL separation(dx,dy,Lx,Ly)
           LET R2(itime) = R2(itime) + dx*dx + dy*dy
           LET Z(itime) = Z(itime) + vx(i)*vxs(i,itime) + vy(i)*vys(itime)
        NEXT i
     NEXT itime
     ! изменяем массивы xs,ys,vxs,vys
     FOR itime = 10 to 2 step -1
        FOR i = 1 to N
           LET xs(itime) = xs(itime-1)
           LET ys(itime) = ys(itime-1)
           LET vxs(itime) = vxs(itime-1)
           LET vys(itime) = vys(itime-1)
        NEXT i
     NEXT itime
     FOR i = 1 to N          ! запоминаем новую конфигурацию
        LET xs(i,1) = x(i)
        LET ys(i,1) = y(i)
        LET vxs(i,1) = vx(i)
        LET vys(i,1) = vy(i)
     NEXT i
  ELSE
```

```
      FOR i = 1 to N          ! запоминаем конфигурацию, если ntrans <=10
         LET xs(i,10 - ntrans) = x(i)
         LET ys(i,10 - ntrans) = y(i)
         LET vxs(i,10 - ntrans) = vx(i)
         LET vys(i,10 - ntrans) = vy(i)
   END IF
   LET ntrans = ntrans + 1          ! количество измерений функций переноса
END SUB

SUB trans_output(N, ntrans,Z(),R2())
   ! нормируются функции переноса и результаты выдаются на печать
   LET ntrans = ntrans - 10
   LET norm = 1/(N* ntrans )
   FOR itime = 1 to 10
      PRINT itime, r2(itime)*norm,z(itime)*norm
   NEXT itime
END SUB
```

ЗАДАЧА 6.10. Коэффициент диффузии

а. Используйте равновесную конфигурацию из задачи 6.5 и визуально проследите движение одной конкретной частицы (называемой *меченой* или *пробной* частицей). Опишите качественно ее движение.

б. Модифицируйте программу **md**, чтобы она рассчитывала коэффициент самодиффузии. Используйте следующую начальную конфигурацию.

```
LET N = 16
LET Lx = 6
LET Ly = 6
DATA 1.5,3.6,0.2,0.5,4.8,5.9,-0.8,-4.6,1.8,5.7,1.8,3.4,1.9
DATA 4.5,0.2,0.5,3.6,4.3,0.3,0.6,0.5,3.2,-3.2,-2.5,5.8,4.0
DATA 3.1,-2.8,3.1,5.8,2.5,-0.1,5.0,2.2,-0.1,0.4,6.0,1.7,2.1
DATA 1.2,4.2,5.2,-0.9,1.9,3.9,3.3,2.1,-0.1,4.3,1.0,-3.0,-1.4
DATA 3.0,0.9,-1.9,2.3,3.0,1.8,-0.0,-0.9,1.5,0.8,-2.3,1.8
FOR i = 1 to N
  READ x(i),y(i),vx(i),vy(i)
NEXT i
```

Возьмите $\Delta t = 0.01$ и вычислите $R(t)^2$ с интервалом в пять шагов

по времени. Кроме того, вычислите средние температуру и давление. Поскольку $R(t)$ гораздо меньше, чем $L_x/2$ или $L_y/2$, влияние конечного размера области сведено к минимуму.

в. Начертите $R(t)^2$ как функцию t. Растет ли $R(t)^2$ линейно с t, т.е. как для свободной частицы или медленнее? Для оценки величины D из графика $R(t)^2$ воспользуйтесь формулой (6.10). Получите D для нескольких разных температур. (При тщательном изучении $R(t)^2$ для систем бóльшего размера и на более продолжительных временах обнаружилось бы, что в двумерном случае величина $R(t)^2$ не пропорциональна t. Вместо этого $R(t)$ имеет член, пропорциональный $t \ln t$, который при достаточно больших t превосходит линейный по t член. Однако мы не сумеем заметить влияние этого логарифмического члена и вправе объяснить свои результаты для $R(t)^2$ на основе «эффективного» коэффициента диффузии. Для трехмерного случая подобной проблемы не существует.)

г. Вычислите $R(t)^2$ для равновесной конфигурации, отвечающей гармоническому твердому телу. Как качественно ведет себя $R(t)^2$? Согласуется ли это поведение с вашими снимками?

д. Вычислите $R(t)^2$ для равновесной конфигурации, отвечающей разреженному газу. Почему величина $R(t)^2$ при малых временах не пропорциональна t? Целесообразно ли использовать молекулярную динамику для вычисления коэффициента самодиффузии разреженного газа?

*ЗАДАЧА 6.11. Автокорреляционая функция скорости

а. Для равновесной конфигурации из задачи 6.10б вычислите $Z(t)$ с интервалом в пять временных шагов. Начертите зависимость $Z(t)$ от t и опишите качественно ее поведение. Оцените D, используя формулу (6.12). (Чтобы вычислить интеграл от $Z(t)$, просуммируйте все свои значения $Z(t)$, полученные при разных значениях t, и умножьте сумму на временной интервал между двумя последовательными значениями t.)

б. Предположим, что зависимость $Z(t)$ от времени описывается формулой $Z(t) \approx A \exp(-t/\tau)$. Чему равно теоретическое значение коэффициента A ? Подставьте эту предполагаемую зависимость $Z(t)$ в формулу (6.12) и определите, как связан коэффициент диффузии D с *временем корреляции* τ. Используйте это полученное соотноше-

ние между D и τ для вывода D. Сравните свои оценки для D, найденные из наклона $R(t)^2$, формулы (6.12) и оценки τ. Совпадают ли эти оценки?

в. Возьмите равновесную конфигурацию из п. «а» и увеличьте плотность, заменив L_x и L_y на 5.5 и пересчитав все координаты в масштабе (5.5/6). После установления в системе равновесия вычислите $Z(t)$ с интервалом в пять шагов по времени. Как качественно ведет себя $Z(t)$? Какой смысл в том, что через относительно непродолжительное время функция $Z(t)$ становится отрицательной? Дайте физическое объяснение этому эффекту.

г. Поместите $N = 16$ частиц в узлы треугольной решетки с $L = 4$, $L = 2\sqrt{3}$, а их скорости выберите случайным образом в интервале $[-1, 1]$. Возьмите $\Delta t = 0.01$ и вычислите $Z(t)$ после того, как будет достигнуто равновесие. Начертите график $Z(t)$ и опишите качественное поведение $Z(t)$. Объясните свои результаты с точки зрения колебательного движения частиц около своих узлов решетки.

6.8. ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ

Основные цели этой главы состояли в том, чтобы ввести некоторые понятия кинетической теории жидкостей и познакомиться с методом молекулярной динамики. Хотя мы и обнаружили, что моделирование систем всего лишь из 16 частиц уже выявляет некоторые качественные свойства макроскопических систем, для получения количественных результатов нам понадобилось бы моделировать системы большого числа частиц. Как правило, больше всего счетного времени занимают формирование равновесного состояния и громоздкая арифметика, необходимая для вычисления силы и энергии. Если радиус действия силы достаточно мал, то имеется ряд способов уменьшить время получения равновесного состояния. Предположим, например, что требуется промоделировать трехмерную систему из 864 частиц. Можно сначала смоделировать меньшую систему из 108 частиц и дать возможность этой малой системе прийти в равновесие при требуемой температуре. После того как равновесие достигнуто, малую систему можно сдублировать в каждом пространственном направлении, с тем чтобы создать искомую систему из 864 частиц. Все скорости переопределяем с помощью распределения Максвелла—Больцмана (см. задачу 6.4), и процедура установления равновесия возобновляется. В полной системе равновесие обычно устанавливается

быстро.

Время счета каждого шага по времени по нашей программе молекулярной динамики в ее настоящем виде примерно пропорционально N^2. Данная зависимость обусловлена тем, что вычисление энергии и сил требует суммирования по всем парам N взаимодействующих частиц. Однако если взаимодействие является короткодействующим, то время, требуемое для подсчета этих сумм на каждом шаге по времени, можно уменьшить до величины порядка N. Идея заключается в использовании того факта, что в любой данный момент времени большинство пар частиц разделяет расстояние гораздо большее, чем эффективный радиус r_c межчастичного потенциала (для потенциала Леннарда—Джонса $r_c \approx \approx 2.55\sigma$). Следовательно, при вычислении силы и энергии нужно учитывать в сумме взаимодействия только тех пар, которые отстоят друг от друга меньше чем на r_c. Разумеется, проверка каждой пары на выполнение этого условия также занимает порядка N^2 операций. Очевидно, что нам нужно ограничить число проверяемых пар. Один метод состоит в том, что ящик разбивается на ряд маленьких ячеек. На каждом шаге по времени составляется список молекул, попавших в каждую ячейку. Эти списки затем просматриваются и относительные расстояния вычисляются только для пар молекул в соседних ячейках.

Как мы узнаем, достаточно ли велик размер системы, чтобы получаемые количественные результаты не зависели от N? Ответ прост — повторяем моделирование с различными N. К счастью, для получения надежных результатов при моделировании равновесных систем с простыми потенциалами в большинстве случаев требуется всего от нескольких сотен до нескольких тысяч частиц.

Метод молекулярной динамики позволяет проводить численные эксперименты при постоянных температуре и/или давлении, а не при постоянных энергии и объеме. Можно также проводить моделирование, в котором форма ячеек определяется динамикой, а не закладывается в программу. Такое моделирование существенно для изучения фазовых переходов в твердом теле из одной твердой фазы в другую, где самым важным является изменение формы кристалла.

Помимо этих методических разработок можно еще многое узнать о свойствах системы из траекторий. Например, как связаны траектории с такими явлениями переноса, как вязкость и теплопроводность? Что представляет собой структура жидкости, например чему равна вероятность того, что две частицы находятся на определенном расстоянии?

Последняя задача будет рассматриваться в гл. 16 в связи с методами Монте-Карло.

Теперь, когда мы знакомы с методом молекулярной динамики, можно кратко обсудить роль молекулярной динамики и других методов моделирования. Как правило, используя методы молекулярной динамики, преследуют в основном две цели. Предположим, что мы хотим разработать общую теорию жидкостей. Чтобы облегчить теоретический анализ, желательно выбрать простую модель, которая сохранит все основные свойства жидкостей. Поскольку такая модель не будет иметь лабораторного аналога, молекулярная динамика может обеспечить по существу точные (квазиэкспериментальные) данные. Кроме того, молекулярная динамика может дать информацию относительно величин, которые с трудом измеряются в лабораторных экспериментах. С другой стороны, моделирование можно было бы сравнить с реальным (физическим) экспериментом для проверки детальных предсказаний теории. Как результат такого типа взаимодействия между теорией и компьютером с лабораторными экспериментами в настоящее время существует вполне развитая теория структур простых жидкостей.

В современных исследованиях по молекулярной динамике основной акцент сдвигается от исследований равновесных жидкостей к исследованиям неравновесных систем. Например, как формируется твердое тело при быстром понижении температуры жидкости? Как распространяется трещина в твердом теле? Численное моделирование будет играть решающую роль в содействии нашему пониманию этих и других проблем.

ЛИТЕРАТУРА

Farid F. Abraham, Computational statistical mechanics: methodology, applications and supercomputing, Adv. Phys. **35**, 1 (1986). Автор рассматривает как метод молекулярной динамики, так и метод Монте-Карло (см. гл. 16). Особое внимание уделяется будущей роли суперкомпьютеров и новых компьютерных архитектур в решении целого спектра задач из самых разных областей науки — от структуры протеинов до турбулентных жидкостей.

B. J. Alder, T. E. Wainwright, Studies in Molecular Dynamics. I. General Method, J. Chem. Phys. **31**, 459 (1960). Хотя мы не рассматривали методы молекулярной динамики для жестких сфер, никакое обсуждение молекулярной динамики не было бы полным без упоминания

этой пионерской работы.

R. P. Bonomo, F. Riggi, The evolution of the speed distribution for a two-dimensional ideal gas: A computer simulation, Am. J. Phys. **52**, 54 (1984). Авторы рассматривают систему жестких дисков и показывают, что система всегда эволюционирует к распределению Максвелла—Больцмана.

Jean-Pierre Hansen, Ian R. McDonald, *Theory of Simple Liquids*, Academic Press, 1976. Отличный справочник, в котором получено большинство теоретических результатов, использованных в данной главе.

J. Kushick, B. J. Berne, Molecular dynamics methods: continuous potentials in: Statistical Mechanics Part B: Time-Dependent Processes, Bruce J. Berne, ed. Plenum Press, 1977. См. также в том же томе статью *Jerome J. Erpenbeck, William Wood,* Molecular dynamics techniques for hard-core systems.

A. Rahman, Correlations in the Motion of Atoms in Liquid Argon, Phys. Rev. **136**, A405 (1964). Первое применение молекулярной динамики к системам с непрерывными потенциалами.

F. Reif, Statistical Physics, Vol. 5 из берклиевского курса по физике, McGraw-Hill, 1965. Введение в физику макроскопических систем, в котором, по-видимому, впервые использовано в целях обучения численное моделирование для иллюстрации приближения макроскопических систем к равновесию.

F. Reif, Fundamentals of Statistical and Thermal Physics, McGraw-Hill, 1965. Учебник средней трудности по статистической механике с более подробным рассмотрением кинетической теории, чем в большинстве книг для младших курсов.

R. M. Sperandeo Mineo, R. Madonia, The equation of state of a hard-particle system: a model experiment on a microcomputer, Eur. J. Phys. **7**, 124 (1986).

Loup Verlet, Computer 'Experiments' on Classical Fluids. I. Thermodynamical Properties of Lennard–Jones Molecules, Phys. Rev. **159**, 98 (1967). Еще одна классическая работа по молекулярной динамике.

James H. Williams, Glenn Joyce, Equilibrium properties of a one-dimensional kinetic system, J. Chem. Phys. **59**, 741 (1973). Моделирование в одномерном случае даже проще, чем в двумерном.

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

Базаров И. П., Геворкян Э. В., Николаев П. Н., Неравновесная термодинамика и физическая кинетика.—М.: МГУ, 1989. В гл. 10 рассматривается метод молекулярной динамики и его приложение к системе частиц с потенциалом Леннарда—Джонса.

Лагарьков А. Н., Сергеев В. М., Метод молекулярной динамики в статистической физике, УФН, 125, 3, 400—448, 1978. В статье подробно рассматриваются вопросы, отраженные в ее названии. Рассчитана на подготовленного читателя.

ПРИЛОЖЕНИЕ 6А. ВИРИАЛ ДАВЛЕНИЯ

Следующий вывод формулы для давления жидкости, заключенной в объеме V, основывается на выводе, приведенном в книге Хансена и Мак-Дональда. Рассмотрим величину

$$G = \sum_i \mathbf{r}_i \cdot \mathbf{F}_i \tag{6.13}$$

где сумма берется по всем частицам системы и $\mathbf{F}_i$ есть полная сила, действующая на i-ю частицу. Среднее по времени от G можно записать в виде

$$\begin{aligned} \langle G \rangle &= \lim_{\tau\to\infty} \frac{1}{\tau} \int_0^\tau dt \sum_i \mathbf{r}_i(t) \cdot \mathbf{F}_i(t) = \\ &= \lim_{\tau\to\infty} \frac{1}{\tau} \int_0^\tau dt \sum_i \mathbf{r}_i(t) \cdot m \frac{d^2 \mathbf{r}_i(t)}{dt^2} = \\ &= -\lim_{\tau\to\infty} \frac{1}{\tau} \int_0^\tau dt \sum_i m \left| \frac{d^2 \mathbf{r}_i(t)}{dt^2} \right|^2 = -3Nk_BT, \end{aligned} \tag{6.14}$$

где мы провели интегрирование по частям и воспользовались теоремой о равнораспределении. G можно разбить на часть, связанную с межчастичными силами, и другую часть, обусловленную внешней силой, действующей со стороны стенок. Последнюю можно связать с давлением и записать

$$d\mathbf{F}_i = -P\hat{n}dA, \tag{6.15}$$

где $\hat{n}$ представляет собой единичный вектор нормали к площадке $d\mathbf{A}$. Имеем

$$\sum_i \mathbf{r}_i \cdot \mathbf{F}_i = -P \int \mathbf{r} \cdot \hat{n}\, dA = P \int \nabla \cdot \mathbf{r}\, dV = 3PV, \tag{6.16}$$

где мы применили теорему Гаусса. В результате можно записать

$$-3PV + \langle G_{int} \rangle = -3Nk_BT$$

или

$$PV = Nk_BT + \frac{1}{3} \left\langle \sum_{i=1} \mathbf{r}_i \cdot \mathbf{F}_i \right\rangle. \tag{6.17}$$

В двумерном случае дробь 1/3 заменяется на 1/2.

ХАОТИЧЕСКОЕ ДВИЖЕНИЕ ДИНАМИЧЕСКИХ СИСТЕМ

7

В этой главе изучаются простые детерминированные нелинейные модели динамических систем, обнаруживающие сложную структуру поведения.

7.1. ВВЕДЕНИЕ

Понятия, которые будут обсуждаться в этой главе, основываются на применении компьютера в качестве инструмента для проведения эмпирических наблюдений. Исследования с помощью компьютера привели к значительному прогрессу в понимании нелинейных явлений.

Большинство явлений природы по сути своей нелинейны. Модели погоды и турбулентный режим движения жидкостей являются общеизвестными примерами нелинейных процессов. Несмотря на то что мы отдаем себе отчет в значимости нелинейных эффектов во многих физических явлениях, некоторые основные понятия легче объяснить, если рассмотреть задачи теоретической экологии. В дальнейшем анализируется разностное уравнение с одной переменной:

$$x_{n+1} = 4r\,x_n(1 - x_n), \tag{7.1}$$

где x_n является отношением численности популяции n-го поколения к численности предыдущего. Мы обнаружим, что динамические свойства уравнения (7.1) чрезвычайно сложны и играют важную роль в разработке более общей теории нелинейных явлений. Важность последних результатов выражается нижеследующей цитатой эколога Роберта Мэя (см. литературу в конце этой главы), относящейся к исследованию разностного уравнения (7.1):

«...Исследование этого уравнения не требует познаний, выходящих за рамки элементарного анализа. Такое исследование развивает интуицию студента, касающуюся нелинейных систем. Не только в науке, но и в повседневной политической и экономической жизни мы делали бы меньше ошибок, если бы большинство людей осознало тот факт, что простые нелинейные системы необязательно обладают простыми динамическими свойствами».

7.2. ПРОСТОЕ ОДНОМЕРНОЕ ОТОБРАЖЕНИЕ

Многие биологические популяции состоят из одного поколения, которое не перекрывается ни с предыдущим, ни с последующим. В качестве примера можно представить некий остров с популяцией насекомых, которые плодятся и откладывют яйца летом, а на следующее лето выводятся новые особи. Поскольку процесс развития популяции дискретен, то более

уместно описывать развитие популяции разностными, а не дифференциальными уравнениями. Простая модель развития популяции, не зависящая от плотности, связывающая численность популяции в $(n+1)$-поколении с предыдущим n-м поколением, записывается в виде

$$P_{n+1} = aP_n, \tag{7.2}$$

где a — некоторая постоянная. Из разностного уравнения следует что, если $a > 1$, то численность каждого поколения будет в a раз больше предыдущего. Это приводит к геометрическому росту и в конце концов к неограниченной численности популяции. Представляется естественным сформулировать более реалистичную модель, в которой численность популяции ограничивается пропускной способностью окружающей среды. Простая дискретная модель прироста популяции, зависящая от плотности, записывается в виде

$$P_{n+1} = P_n(a - bP_n). \tag{7.3}$$

Заметим, что выражение (7.3), иногда называемое «логистическим» разностным уравнением, нелинейно из-за наличия квадратичного члена P_n. Первый член представляет естественный прирост численности популяции; квадратичный член представляет уменьшение естественного прироста, например, за счет перенаселения или распространения болезней.

Для удобства «перемасштабируем» численность популяции, положив $P_n = (a/b)x_n$ и переписав уравнение (7.3) в виде

$$x_{n+1} = ax_n(1 - x_n). \tag{7.4}$$

Переход от переменной P_n к x_n изменяет систему единиц, используемую для определения различных параметров. Чтобы записать уравнение (7.4) в стандартном виде (7.1), введем параметр «роста» $r = a/4$ и получим

$$x_{n+1} = f(x_n), \tag{7.5}$$

где функция $f(x_n)$ записывается в виде

$$f(x) = 4rx(1 - x). \tag{7.6}$$

Функция, записанная в новых переменных (7.6), обладает желательным свойством, которое заключается в том, что ее поведение определяется единственным параметром r, который можно менять. Заметим, что если $x_n > 1$, то значение x_{n+1} будет отрицательным. Чтобы избежать этой нефизической ситуации, наложим условия, которые ограничивают изменение переменной x и параметра r отрезками $0 \le x \le 1$ и $0 \le r \le 1$. Поскольку функция $f(x)$ переводит любую точку отрезка в некоторую

ТАБЛИЦА 7.1. Первые шестнадцать итераций отображения $x_{n+1} = 4rx_n(1 - x_n)$ для четырех значений параметра r. Обратите внимание на то, что в каждом случае устойчивые состояния различны

n	$r = 0.1$	0.6	0.8	0.9
0	0.750000	0.750000	0.750000	0.750000
1	0.075000	0.450000	0.600000	0.675000
2	0.027750	0.594000	0.768000	0.789750
3	0.010792	0.578794	0.570163	0.597762
4	0.004270	0.585100	0.784247	0.865593
5	0.001701	0.582619	0.541452	0.418829
6	0.000679	0.583618	0.794502	0.876281
7	0.000271	0.583219	0.522460	0.390286
8	0.000109	0.583379	0.798386	0.856667
9	0.000043	0.583515	0.515091	0.442040
10	0.000017	0.583341	0.799271	0.887906
11	0.000007	0.583330	0.513398	0.358303
12	0.000003	0.583334	0.799426	0.827719
13	0.000001	0.583333	0.513102	0.513360
14	0.000000	0.583334	0.799451	0.899357
15	0.000000	0.583333	0.513054	0.325849
16	0.000000	0.583333	0.799455	0.790817

другую точку того же самого отрезка, функция $f(x)$ называется одномерным *отображением*. В дальнейшем мы будем ссылаться на уравнение (7.6) для функции $f(x)$ как на «стандартное» отображение.

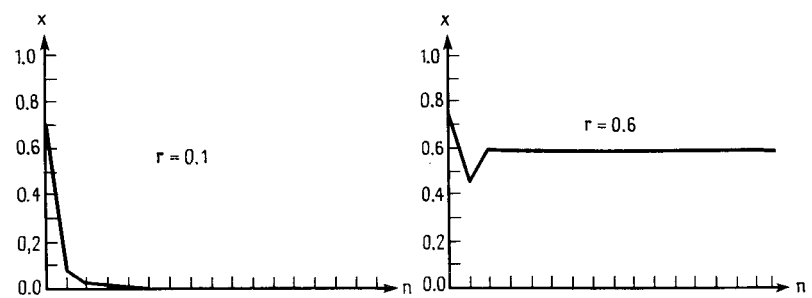

Рис. 7.1, *а*. График итерированных значений x в зависимости от номера итерации для случаев $r = 0.1$ и $r = 0.6$.

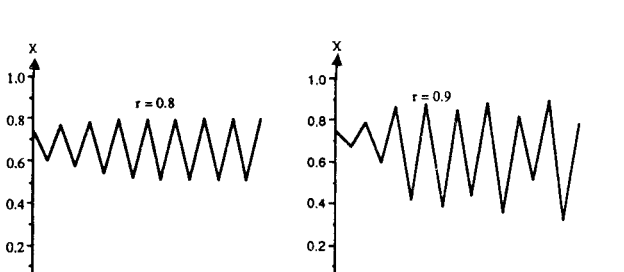

Рис. 7.1, *б*. График итерированных значений x в зависимости от номера итерации для случаев $r = 0.8$ и $r = 0.9$.

Много интересных свойств стандартного отображения были открыты Фейгенбаумом в 1978г. с помощью программируемого калькулятора. В табл. 7.1 приводятся первые шестнадцать итераций для различных значений параметра r, но для одинакового начального значения переменной $x_0 = 0.75$, указывающие на различное поведение отображения $f(x)$. Эти данные также представлены в виде графиков рис. 7.1. Заметим,

что несколько начальных итераций отображения $f(x)$ ведут себя странным образом, но тем не менее, за исключением случая $r = 0.9$, проявляется некая закономерность. Начальный отрезок последовательности называется переходным режимом, а остальная часть является установившимся режимом. Последовательность значений x_n называется *орбитой* отображения.

В приведенных ниже программах вычисляется орбита одномерного отображения $f(x)$. В программе **map_table** значения x_n распечатываются в формате восемь итераций в строке. Значение r можно изменить нажатием любой клавиши. В программе **map_plot** строится график x в зависимости от номера итерации n. В процессе построения графиков можно изменить значение r на некоторую величину Δr, нажав клавиши 'i'(увеличение) или 'd'(уменьшение).

```
PROGRAM map_table      ! итерации одномерного отображения f(x)
DO
 CALL initial(x,r)
 CALL map(x,r)
 INPUT prompt "продолжать? (y/n - да/нет)":choice$
LOOP until choice$ = "n"
END

SUB initial(x,r)
 INPUT prompt "параметр роста (0 < r < 1) = ":r
 INPUT prompt "начальное значение x (0 < x < 1) = ":x
END SUB

SUB map(x,r)
 LET iterations = 0
 DO
  LET x = 4*r*x*(1 - x)
  LET iterations = iterations + 1              ! номер итерации
  PRINT USING "*.******": x;
  IF mod(iterations,8) = 0 then PRINT          ! новая строка
LOOP until key input
PRINT
PRINT "число итераций = "; iterations
END SUB
```

```
PROGRAM map_plot
CALL initial(x, r, nmax)
CALL plot_output(x, r, nmax)
END

SUB initial(x, r, nmax)
 INPUT prompt "начальное значение параметра роста r = ": r
 INPUT prompt "начальное значение x = ": x
 INPUT prompt "максимальное количество итераций = ": nmax
 SET WINDOW -.1*nmax, nmax, -.1, 1.1
 PLOT LINES: 0,1; 0,0; nmax,0                    ! рисование осей x и y
END SUB

SUB plot_output(x, r, nmax)
 DECLARE DEF f
 LET dr = 0.01                                   ! приращение r
 FOR i = 1 to nmax
  IF key input then
  ! нажатие клавиши 'i' увеличивает r на dr, а 'd' уменьшает r
      GET KEY k
      IF k = ord("i") then LET  r = r + dr
      IF k = ord("d") then LET  r = r - dr
      SET cursor 1,1
      PRINT using "r = **.*****": r
  END IF
  LET x = f(x, r)
  PLOT POINTS: i, x
 NEXT i
END SUB

DEF f(x, r) = 4*r*x*(1 - x)
```

Для выяснения динамических свойств стандартного отображения (7.6) в задачах 7.1 и 7.2 используются программы **map_table** и **map_plot**.

ЗАДАЧА 7.1. Исследование удвоения периода

а. Изучите динамическое поведение стандартного отображения (7.6) для значений параметра $r = 0.2$ и $r = 0.24$ и для различных значений *источника* x_0. Покажите, что $x = 0$ является *устойчивой непо-*

движной точкой. Иначе говоря, для достаточно малого значения параметра r итерации x_n сходятся к $x = 0$ независимо от начального условия x_0. В том случае, когда переменная представляет собой численность популяции насекомых, качественно охарактеризуйте динамику этой популяции.

б. Исследуйте динамическое поведение стандартного отображения (7.6) для значений параметра $r = 0.26$, 0.5, 0.7, 0.72, 0.74 и 0.748. (В случае $r = 0.748$ для сходимости итерационного процесса необходимо приблизительно 1000 итераций.) Сходится ли процесс к значению $x = 0$? Неподвижная точка называется *неустойчивой*, если для почти всех значений x_0 итерационный процесс расходится. Свидетельствуют ли ваши результаты о том, что $x = 0$ — неустойчивая неподвижная точка? Покажите, что через много поколений итерированные значения переменной x постоянны, т.е. динамический режим является стационарным или имеет *период*, равный 1. Каковы устойчивые неподвижные точки для различных значений параметра r? Последовательность итераций x_0, x_1, x_n ... называется *орбитой*, или *траекторией* x. Покажите, что для любого из предложенных значений параметра r орбиты x по прошествии начального переходного периода не зависят от начального значения.

в. Исследуйте динамическое поведение стандартного отображения (7.6) для значений параметра $r = 0.752$, 0.76, 0.8 и 0.862. (В случае $r = 0.752$ для сходимости итерационного процесса необходимо приблизительно 1000 итераций.) Покажите, что если параметр становится чуть больше 0.75, то после переходного режима x осциллирует между двумя значениями, т.е. вместо устойчивого цикла с периодом, равным 1, соответствующего одной неподвижной точке, у системы имеется устойчивый цикл с периодом 2. Значение параметра r, при котором единственная неподвижная точка x^* расщепляется, или происходит *бифуркация* на два осциллирующих значения x_1^* и x_2^*, равно $r = 3/4$. Пара величин (x_1^* и x_2^*) образует *устойчивый аттрактор* с периодом 2.

г. Опишите экологический сценарий популяции насекомых или человеческого общества, которые ведут себя аналогично отображению из п. «в».

д. Что является устойчивым аттрактором стандартного отображения

(7.6) для значений параметра r = 0.863 и 0.88? Чему равен период в каждом случае?

е. Что является устойчивым аттрактором стандартного отображения и чему равны соответствующие периоды для значений параметра r = = 0.89, 0.891 и 0.8922?

ЗАДАЧА 7.2. Хаотический режим

а. Область значений параметра $r > r_c$ = 0.892486417967... называется *хаотическим* режимом, в котором две близлежащие начальные точки разбегаются по различным траекториям после небольшого числа итераций. В качестве примера выберите такие источники x_0 = 0.500 и 0.501. Сколько итераций необходимо для того, чтобы последующие значения различались между собой более чем на 10%?

б. Известно, что точность представления чисел с плавающей запятой в компьютере конечна. Для проверки влияния конечной точности вашего компьютера выберите сначала значения r = 0.91 и x_0 = 0.5 и получите численное значение x после приблизительно 200 итераций. Затем модифицируйте свою программу так, чтобы последовательно выполнялись операции $x = x/10$ и $x = 10*x$. Эта комбинация действий обрезает последнюю десятичную цифру, которую хранит компьютер. (Аналогичного эффекта можно добиться с помощью функ-

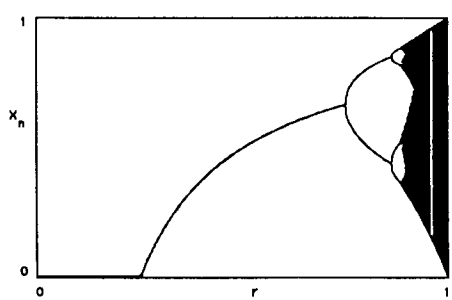

Рис. 7.2. График итерированных значений x_n в зависимости от параметра роста r. Обратите внимание на переход от периодического движения к хаотическому. Обратите также внимание на узкие окна периодического движения внутри областей хаоса.

ции **truncate** в языке True BASIC.) Получите итерированное значение x при тех же условиях и сравните результаты. Будет ли такое же несовпадение и в случае $r < r_c$?

в. Каковы динамические свойства системы для значения параметра $r = 0.958$. Можете ли вы найти другие «окна» в этом хаотическом режиме?

Программа **map_table** печатает, а программа **map_plot** рисует график значений численности популяции x в зависимости от числа итераций. Другой способ представить поведение (7.6) заключается в построении графика зависимости x от управляющего параметра r (рис. 7.2). На рис. 7.2 нанесены итерированные значения x, полученные только после завершения переходного периода. Такой график можно получить, модифицируя программу **map_plot** таким образом, чтобы для заданного значения r первые *ntransient* значений x вычислялись, но не выводились на график, а следующие *nplot* значений вычислялись с нанесением на график. Эта процедура повторяется для нового значения параметра r до тех пор, пока не достигается желаемый диапазон значений параметра r. Типичное значение *ntransient* должно лежать в диапазоне 100—1000 итераций. Типичное значение *nplot* должно быть по крайней мере не меньше, чем самый большой период, который вы хотите увидеть.

ЗАДАЧА 7.3. Качественные особенности квадратичного отображения

а. Модифицируйте программу **map_plot** так, чтобы итерации x_n строились в виде графика, зависящего от r. Сделайте диапазоны величин r и x входными параметрами и измените инструкцию **SET window** так, чтобы вы могли «увеличивать масштаб изображения» любой части графика, приведенного на рис. 7.2. Не нужно наносить на график первые *nplot* итераций. Начните с диапазона $0.8 \le r \le 0.9$. Сколько удвоений периода вы можете различить?

б. Измените масштаб так, чтобы вы могли наблюдать итерации x от периода 4 до периода 32. Как выглядит график в этом масштабе по сравнению с графиком для отображения с периодом 4 в исходном масштабе?

в. Дайте краткое качественное описание поведения кривой вблизи точек бифуркации.

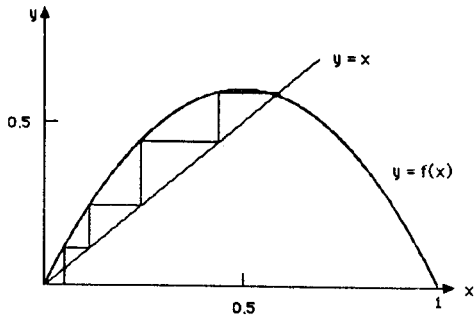

Рис. 7.3. Итерации отображения $x_{n+1} = 4rx_n(1 - x_n)$ с $r = 0.6$ и начальным значением $x_0 = 0.05$. Неподвижная точка $x = 0$ — неустойчива, а неподвижная точка $x = 0.58333$ является устойчивой.

7.3. УДВОЕНИЕ ПЕРИОДА

Приведенные выше «машинные эксперименты», касающиеся поведения стандартного отображения, привели к созданию нового словаря для описания наших наблюдений и, вероятно, убедили вас в том, что свойства простых динамических систем могут быть очень сложными!

Для понимания зависимости динамического поведения от параметра r представим простой и элегантный графический метод итерирования $f(x)$. На рис. 7.3 приведен график $f(x)$ для значения параметра $r = 0.6$. Наклонная прямая, соответствующая функции $y = x$, пересекает кривую $y = f(x)$ в двух неподвижных точках $x^* = 0$ и $x^* = 0.58333$, т.е. повторение итераций функции $f(x)$ для значений $x^* = 0$ и $x^* = 0.58333$ дает постоянную последовательность. Если x_0 не является одной из неподвижных точек, мы можем найти орбиту следующим образом. Сначала проводим вертикальную прямую из точки $\{x = x_0, y = 0\}$ до пересечения с кривой $y = f(x)$ в точке $\{x_0, y_0 = f(x_0)\}$. Затем проводим горизонтальную прямую из точки $\{x_0, y_0\}$ до пересечения с наклонной прямой в точке $\{y_0, y_0\}$. Поскольку на этой наклонной прямой значение y равно значению x, то значение x в точке пересечения является первой итерацией $x_1 = y_0$. Аналогично можно найти вторую итерацию x_2. Из точки $\{x_1, y_0\}$ проводим вертикальную прямую до пе-

ресечения с кривой $y = f(x)$. Фиксируем точку $y = y_1 = f(x_1)$ и проводим горизонтальную прямую до пересечения с наклонной; значение x в точке пересечения дает x_2. Дальнейшие итерации можно найти, повторяя следующую процедуру:

1) двигаемся по вертикали до пересечения с кривой $y = f(x)$;
2) двигаемся по горизонтали до пересечения с наклонной прямой $y = x$;
3) повторяем шаги 1 и 2 бесконечное число раз.

Этот графический метод иллюстрируется на рис. 7.3 для $x_0 = 0.05$ и $r = 0.6$. Заметим, что если начинать итерации из любой точки x_0 ($x \neq 0$ и $x \neq 1$), то итерационный процесс будет сходиться к неподвижной точке $x^* = 7/12 \approx 0.58333$. (С помощью карандаша проверьте этот факт на рис. 7.3.) Такие неподвижные точки называются устойчивыми (аттрактор с периодом 1). По сравнению с этим независимо от близости x_0 к неподвижной точке $x = 0$ итерационный процесс будет расходиться. Такая неподвижная точка называется неустойчивой.

Как можно объяснить качественное различие между неподвижными точками $x^* = 0$ и $x^* = 0.58333$ для $r = 0.6$? Локальная кривизна кривой $y = f(x)$ определяет горизонтальное смещение при каждой итерации f. Крутой наклон (более 45°) приводит к удалению x от начального значения. Следовательно, критерий устойчивости неподвижной точки заключается в том, что величина наклона в неподвижной точке должна быть менее 45°, т.е. если $|df(x)/dx|_{x=x^*} < 1$, то точка x^* является устойчивой, и, наоборот, если $|df(x)/dx|_{x=x^*} > 1$, тогда точка x^* неустойчива. Внимательное изучение функции $f(x)$, изображенной на рис. 7.3, показывает, что $x = 0$ — неустойчивая точка, поскольку тангенс угла наклона кривой $f(x)$ в точке $x = 0$ больше единицы. В противоположность этому значение производной $f(x)$ в точке $x = 0.58333$ меньше единицы. В приложении 7А, используя аналогичные соображения, мы показываем аналитически, что

$$x^* = 0 \text{ } \textit{устойчива для } 0 < r < 1/4 \tag{7.7а}$$

и

$$x^* = 1 - \frac{1}{4r} \text{ } \textit{неустойчива для } 1/4 < r < 3/4. \tag{7.7б}$$

Таким образом, для значений $0 < r < 3/4$ конечное поведение известно.

Что происходит, если r лежит в интервале $3/4 < r < 1$? Нам известно из наблюдений, что по мере роста r неподвижная точка функции f становится неустойчивой и приводит к рождению (бифуркации) цикла с периодом 2. Теперь только после каждой второй итерации x принимает то же самое значение, т.е.

$$x_i = f(f(x_i)) \qquad i = 1, 2 \tag{7.8}$$

и аттракторы функции $f(x)$ являются неподвижными точками функции $g(x) = f(f(x))$. Что произойдет, если дальше увеличивать значение параметра r? В конце концов величина наклона неподвижных точек $g(x)$ достигнет единичного значения и неподвижные точки удвоятся. Теперь период f равен 4 и можно изучать устойчивость неподвижных точек четырежды итерированной функции $h(x) = g(g(x)) = f(f(f(f(x))))$. Эти неподвижные точки также в конце концов удваиваются и наблюдается явление *удвоения периода*, т.е. период 1 $\longrightarrow$ период 2 $\longrightarrow$ период 4 $\longrightarrow$ период 8 $\longrightarrow$ период 16 $\longrightarrow$ период 32 $\longrightarrow$ ..., что мы и видели в задаче 7.4.

В программе **map_graph** применяется графический анализ $f(x)$. Обратите внимание на то, что в описании функции $f(x, r, iterate)$ вторая итерация $g(x) = f(f(x))$ и четвертая итерация $h(x) = g(g(x)) = f(f(f(f(x))))$ определяются с помощью рекурсии, т.е. обращения функции к себе самой. (Значение *iterate* равно 1, 2 и 4 для функций $f(x)$, $g(x)$ и $h(x)$ соответственно.)

```
PROGRAM map_graph
CALL parameter(x, r, iterate)
CALL graph(r, iterate)
CALL map(x, r, iterate)
END

SUB parameter(x, r, iterate)
  INPUT prompt " r = ": r
  INPUT prompt "начальный x = ": x
  INPUT prompt "итерации f(x) = ": iterate
END SUB
```

```
SUB graph(r,iterate)
 DECLARE DEF f
 LET n = 200          ! число точек, в которых вычисляется функция
 LET delta = 1/n
 LET margin = 0.1
 SET window -margin,1 + margin,-margin,1 + margin
 PLOT LINES: 0,0;1,1          ! построение прямой y = x
 PLOT LINES: 0,1;0,0;1,0      ! построение осей
 PLOT
 FOR i = 1 to n
     LET x = x + delta
     LET y = f(x,r,iterate)
     PLOT x,y
 NEXT i
END SUB

SUB map(x,r,iterate)
 DECLARE DEF f
 LET n = 100                  ! число итераций отображения
 LET y0 = 0
 LET x0 = x
 FOR i = 1 to n
    LET y = f(x,r,iterate)
    PLOT LINES: x0,y0; x0,y; y,y
    LET x0 = y
    LET y0 = y
    LET x = y
 NEXT i
END SUB

DEF f(x,r,iterate)            ! f определяется с помощью рекурсии
  IF iterate > 1 then
    LET y = f(x,r,iterate - 1)
    LET f = 4*r*y*(1 - y)
  ELSE
    LET f = 4*r*x*(1 - x)
  END IF
END DEF
```

В задаче 7.4 используется программа **map_graph** для исследования динамических свойств стандартного отображения и итераций этого отображения более высокого порядка.

ЗАДАЧА 7.4. Качественные свойства неподвижных точек

а. Используйте программу **map_graph** и покажите графически, что при $r < 3/4$ у функции $f(x)$ существует единственная устойчивая неподвижная точка. Функция $f(x)$ четна относительно точки $x = 1/2$, в которой $f(x)$ имеет максимум. Каковы качественные особенности второй итерации этой функции $g(x) = f(f(x))$? Четна ли функция $g(x)$ относительно точки $x = 1/2$? Является ли точка x^* также неподвижной точкой функции $g(x)$? При каком значении x функция $g(x)$ имеет минимум? Пусть r_1 — значение r, при котором неподвижная точка функции $f(x)$ становится неустойчивой. Убедитесь в том, что $r_1 = -0.75$.

б. Охарактеризуйте орбиту функции $f(x)$ для значения параметра $r = 0.785$. Чему равен период $f(x)$? Чему равны численные значения неустойчивых аттракторов? Проитерируйте $g(x)$ и найдите две неподвижные точки x_1^* и x_2^* этого отображения. (Попробуйте два начальных значения $x_0 = 0.1$ и $x_0 = 0.3$.) Являются ли неподвижные точки отображения $g(x)$ устойчивыми или неустойчивыми? Как соотносятся значения x_1^* и x_2^* со значениями неустойчивых аттракторов отображения $f(x)$? Убедитесь в том, что наклоны функции $g(x)$ в точках x_1^* и x_2^* одинаковы.

в. Проверьте следующие свойства неподвижных точек отображения $g(x)$. По мере увеличения параметра r неподвижные точки $g(x)$ расходятся и наклон $g(x)$ в неподвижных точках уменьшается. При каком значении $r = r^{(1)}$ одна из неподвижных точек g равна $1/2$? Чему равно значение другой неподвижной точки? При этом значении параметра r наклон в обеих неподвижных точках равен нулю. При дальнейшем увеличении r наклоны в неподвижных точках становятся отрицательными. Наконец, при $r = r_2 \approx 0.8623$ наклоны в обеих неподвижных точках функции $g(x)$ становятся равными -1 и эти две неподвижные точки превращаются в неустойчивые.

г. Покажите, что для значений r, чуть больших r_2, например $r \approx 0.87$, у функции $h(x) = g(g(x))$ имеются четыре неподвижные точ-

ки. Чему равно значение $r = r^{(2)}$, при котором одна из неподвижных точек равна 1/2? Чему равны значения трех остальных неподвижных точек при $r = r^{(2)}$? При каком значении $r = r_3$ четыре неподвижные точки отображения h становятся неустойчивыми?

7.4. УНИВЕРСАЛЬНЫЕ СВОЙСТВА НЕЛИНЕЙНЫХ ОТОБРАЖЕНИЙ

До сих пор мы изучали простое отображение (7.6). Для выяснения общих закономерностей удвоения периода рассмотрим в задаче 7.5 два других одномерных отображения.

ЗАДАЧА 7.5. Другие одномерные отображения

Проведите численные эксперименты для определения качественных свойств отображений

$$f(x) = xe^{r(1-x)}, \tag{7.9}$$

$$f(x) = [1 - (2x - 1)^4]. \tag{7.10}$$

Как ведет себя (7.9) для $r \sim 2$ и $r \sim 2.7$? Заметим, что для отображения (7.10) r и начальное значение x должны лежать между 0 и 1. Проявляют ли эти отображения сходные качественные свойства, например удвоение периода и существование хаотической области? Отображение (7.9) использовалось экологами (Мэй) для изучения популяции, ограниченной при больших плотностях эпидемиями. Несмотря на то что данное отображение является более сложным по сравнению с (7.6), у него имеется одно важное преимущество, состоящее в том, что численность популяции всегда положительна независимо от выбираемого начального значения. Не существует никаких ограничений на максимальное значение r, но если величина r становится достаточно большой, то x в итоге будет фактически равняться нулю, следовательно, популяция вымирает.

В дальнейшем мы убедимся в том, что удобно определять «порядок» отображения. Пусть x_{max} — значение, при котором $f(x)$ достигает максимума, т.е. $df/dx = 0$ при $x = x_{max}$. Если при $x = x_{max}$ производные $d^m f/dx^m = 0$ для $m < n$, а $d^n f/dx^n < 0$, то $f(x)$ является отображением

n-го порядка. Покажите, что этот критерий означает, что отображения (7.6) и (7.9) являются отображениями второго порядка или квадратичными. Покажите, что по сравнению с этим отображение (7.10) имеет четвертый порядок, т.е. $d^2f/dx^2 = d^3f/dx^3 = 0$, а $d^4f/dx^4 < 0$ при $x = 1/2$.

Приведем качественные соображения, свидетельствующие о том, что некоторые количественные свойства стандартного квадратичного отображения (7.6) в режиме удвоения периода присущи также всем квадратичным отображениям. Рассмотрим $f(x,r)$ при таком значении r, что период $f(x,r)$ равен 4 (рис. 7.4).

Теперь рассмотрим вторую итерацию $g(x,r) = f(f(x,r))$ для того же самого значения r (рис. 7.5). Ясно, что $g(x,r)$ имеет период 2 и при начальном значении $x_0 = 0.5$ осциллирует между обеими неподвижными точками $x_1^* = 0.373$ и $x_2^* = 0.512$. Можно видеть, что $g(x,r)$ качественно ведет себя подобно $f(x,r')$, где r' — меньшее значение r, для которого $f(x,r')$ имеет период 2. На рис. 7.6 показана $f(x,r')$ для произвольно выбранного значения $r' = 0.8$. Заметим, что $f(x,r = 0.8)$ колеблется между значениями 0.513 и 0.799. Теперь сравним форму $g(x,r = 0.88)$ внутри циркуляционного прямоугольника на рис. 7.5 и форму $f(x,r' = 0.8)$ внутри циркуляционного прямоугольника, показанного на рис. 7.6. Мы видим, что если повернуть g вокруг горизонтальной оси, проходящей через центр прямоугольника, а затем *увеличить* g так, чтобы циркуляциионные прямоугольники для f и g имели одинаковые размеры, то обе функции внутри прямоугольников будут качественно одинаковы. Можно определить коэффициент увеличения α, заметив, что диапазон изменения $g(x,r = 0.88)$ равен $0.512 - 0.373 = 0.139$, а диапазон изменения $f(x,r = 0.8)$ равен $0.799 - 0.513 = 0.286$. Следовательно, если выбрать $\alpha = 0.286/0.139 = 2.06$, то обе функции будут вести себя сходным образом. Эта процедура иллюстрируется на рис. 7.7, где наложены циркуляционные прямоугольники функций $f(x,r = 0.8)$ и $g(x,r = 0.88)$.

Рассмотренный выше множитель представляет собой пример *масштабного* множителя; чтобы сравнить g с f, мы отмасштабировали g и изменили (*перенормировали*) значение r. (См. гл. 12, в которой масштабирование и перенормировка обсуждаются в другом контексте.) Наши доводы были наводящими. Например, мы не объяснили выбор $r' = 0.8$. Оказывается, что достаточно сравнить циркуляционные прямоугольники для значений r, соответствующих неподвижной точке $x = 1/2$ и неподвижной точке, ближайшей к $x = 1/2$. Более строгий подход показывает,

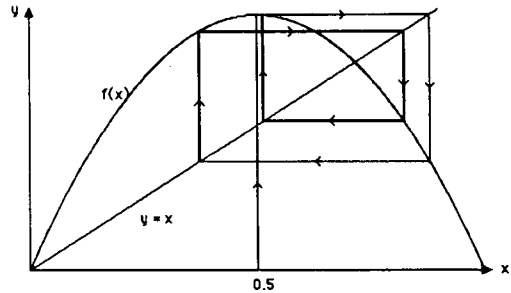

Рис. 7.4. Итерации $f(x)$ для $r = 0.88$. Заметьте, что для этого значения r функция $f(x)$ имеет период, равный 4.

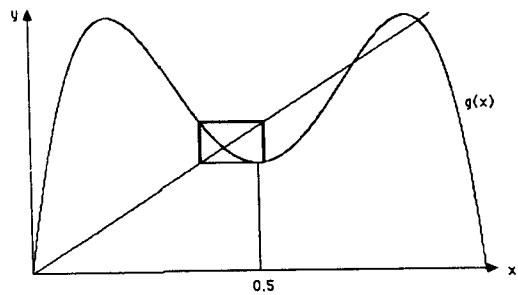

Рис. 7.5. Итерации $g(x)$ для $r = 0.88$, стартующие из $x_0 = 0.5$. Обратите внимание на форму $g(x)$ внутри «циркуляционного прямоугольника», ограниченного значениями $x_1^* = 0.373$ и $x_2^* = 0.512$.

что если продолжить сравнение итераций более высокого порядка, например $h(x)$ с $g(x)$ и т.д., то для всех отображений одного и того же порядка совмещение функций будет сходиться к некой универсальной функции, не зависящей от вида начальной функции $f(x)$.

Число α можно определить из родственных соображений. Посмотрите на «камертонные» бифуркации, изображенные на рис. 7.3 и 7.8. Заметим, что каждый камертон порождает «двойню» новых поколений, более плотно упакованных по сравнению с предыдущим поколением. Для количественной характеристики роста плотности неподвижных точек рассмотрим поведение функции $f(x)$ при значениях $r = r^{(n)}$, для которых одна из неподвижных точек равна 1/2. Например, в задаче 7.4 мы нашли $r^{(1)} \approx \approx 0.809$ и $r^{(2)} \approx 0.875$. Одной из мер плотности может служить величина $d_n = x_n^* - 1/2$, где x_n^* — значение неподвижной точки,

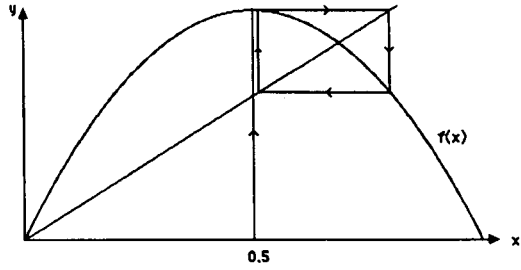

Рис. 7.6. Двухпериодическое поведение функции $f(x)$ для $r = 0.8$. Сравните вид функции $f(x)$ в циркуляционном прямоугольнике, ограниченном значениями $x_1^* = 0.513$ и $x_2^* = 0.799$, с поведением функции $g(x)$ в циркуляционном прямоугольнике, показанном на рис. 7.5.

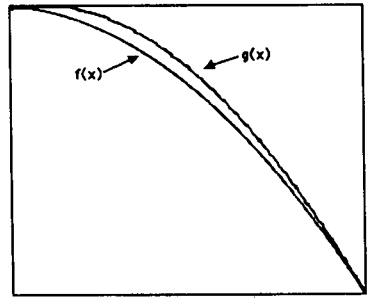

Рис. 7.7. Наложение соответственно увеличенных циркуляционных прямоугольников для функций $f(x, r = 0.8)$ и $g(x, r = 0.88)$.

ближайшей к неподвижной точке $x^* = 1/2$. Первые два значения d_n показаны на рис. 7.8, где $d_1 \approx 0.309$ и $d_2 \approx -0.117$. Заметим, что неподвижная точка, ближайшая к $x = 1/2$, переходит с одной стороны прямой $x = 1/2$ на другую. Определим величину α с помощью отношения

$$\alpha = \lim_{n\to\infty} -\frac{d_n}{d_{n+1}}. \qquad (7.11)$$

Оценка $\alpha = 0.309/0.117 = 2.64$ согласуется с нашими предыдущими оценками и асимптотическим пределом $\alpha = 2.502907875095892 8485...$ (Целый ряд десятичных цифр приведен для того, чтобы показать, что это число известно с большой точностью!)

Мы можем количественно описать процесс удвоения периода с последующим переходом к хаосу. Напомним, что r_n является значением r, при котором впервые появляются 2^n циклов. В задаче 7.4 мы нашли значения $r_1 = 0.75$, $r_2 \approx 0.862$ и $r_3 \approx 0.880$. Фейгенбаум показал (с помощью калькулятора), что по мере роста n значение r_n приближается к предельному значению r_c по простому закону:

$$r_n - r_c = A\delta^{-n}. \tag{7.12}$$

Замечательный результат работы Фейгенбаума заключается в том, что постоянная δ, как и α, является *универсальной*, т.е. δ не зависит от детальных свойств $f(x)$, а зависит только от порядка отображения. Напротив, постоянная A зависит от детальной структуры функции $f(x)$. Из формулы (7.12) непосредственно выводится, что δ можно также определить с помощью соотношения

$$\delta = \lim_{n\to\infty} \frac{r_{n+1} - r_n}{r_{n+2} - r_{n+1}}. \tag{7.13}$$

Используя приведенные выше значения r_1, r_2 и r_3, получим $\delta \approx 6.22$; асимптотическое значение равно $\delta = 4.66920160910299097...$.

ЗАДАЧА 7.6. Дальнейшие оценки универсальных постоянных α и δ

а. Пользуясь рассуждениями, аналогичными тем, которые приведены в тексте, сравните поведение $g(x)$ в циркуляционном прямоугольнике при $x = 1/2$ для $r = r^{(2)} \approx 0.875$ с поведением $f(x)$ в соответствующем циркуляционном прямоугольнике для $r^{(1)} \approx 0.809$. Найдите соответствующий масштабный множитель α и наложите f на перемасштабированную функцию g.

б. Проведите анализ, аналогичный п. «а», и сравните $h(x)$ при $r = r^{(3)} \approx 0.880$ с $g(x)$ при $r = r^{(2)}$. Определите масштабный множитель и наложите эти две функции.

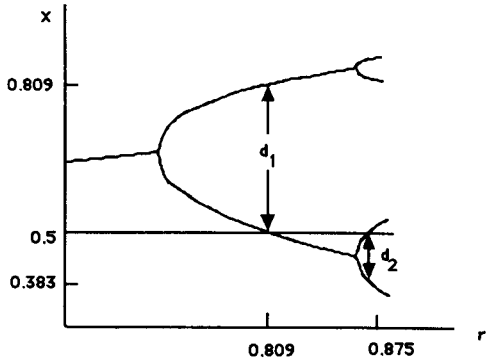

Рис. 7.8. Зависимость x от параметра роста r. Значение d_n является расстоянием от $x^* = 1/2$ до ближайшего элемента аттрактора с периодом 2^n.

в. Вычислите значения r_n для процессов удвоения периода $8 \longrightarrow 16$ и $16 \longrightarrow 32$. Используйте эти значения r_n для улучшения оценки δ.

г. Получите дополнительные значения для $r^{(n)}$, значений r, при которых 2^n итераций функции $f(x)$ имеют неподвижную точку при $x^* = 1/2$. Получите дополнительные оценки δ, воспользовавшись выражением (7.13) с заменой r_n на $r^{(n)}$.

*д. Вычислите α и δ для отображений (7.9) и (7.10).

Из приведенных выше рассуждений можно сделать вывод о том, что существуют *универсальные* постоянные α и δ, не зависящие от конкретного вида $f(x)$. Почему универсальность бифуркаций и существование универсальных постоянных относятся к чрезвычайно редким явлениям? Одна из причин заключается в том, что едва ли популяция будет эволюционировать в точном соответствии с отображением (7.6) или любым другим из ранее рассмотренных нами отображений. Однако если поведение не зависит от деталей функции, описывающей его, то могли бы существовать реальные системы, динамика которых была бы аналогична поведению простых отображений, которые мы рассмотрели. Если бы динамика была аналогичной, то мы бы узнали, что динамику системы со многими степенями свободы можно при определенных условиях упростить.

Конечно, физические системы обычно описываются дифференциальны-

ми, а не разностными уравнениями. Может ли происходить в таких системах удвоение периода? Некоторые исследователи (Теста и др.) сконструировали нелинейную *RLC*-цепь, присоединенную к генератору гармонического напряжения. Выходное напряжение испытывало бифуркации, а измеренные значения α и δ согласовываются со значениями, предсказанными для простого квадратичного отображения.

Поскольку электрические цепи можно описать с помощью нескольких переменных, то не могло вызвать удивления, что они ведут себя подобным образом. Более интересный случай представляет собой природа турбулентности в жидкости, которая является одной из главных областей исследования учеными различных специальностей. Рассмотрим, например, поток воды, обтекающий несколько препятствий. Из опыта известно, что при малых скоростях поток является регулярным и постоянным во времени и называется *ламинарным* течением. По мере роста скорости потока (определяемой с помощью безразмерного параметра, который называется числом Рейнольдса) появляются завихрения, но движение все еще постоянно. Если еще более увеличить скорость потока, то вихри опрокидываются и начинается движение нижних слоев. Поток, который мы наблюдаем, находясь на уступе, становится нестационарным. Если и дальше скорость увеличивается, то поток становится очень сложным и выглядит хаотическим. Мы говорим, что течение воды совершило переход из ламинарного режима в *турбулентный*.

Это качественное описание перехода к хаосу в гидродинамических системах внешне выглядит аналогично описанию простого квадратичного отображения. Можно ли гидродинамические системы проанализировать с помощью простых моделей такого типа, которые мы здесь обсуждали? В некоторых частных случаях, таких как турбулентная конвекция в подогретом соуснике, удвоение периода и других, наблюдались переходы в турбулентный режим. Вообще такого типа теория и анализ, который мы провели, породили новые концепции и подходы. Тем не менее истинное представление природы турбулентных потоков остается предметом многих современных исследований.

*7.5. ХАОТИЧЕСКОЕ ПОВЕДЕНИЕ В КЛАССИЧЕСКОЙ МЕХАНИКЕ

Поскольку по крайней мере некоторые качественные особенности квадратичного отображения наблюдаются в лабораторных условиях, интересно рассмотреть динамические свойства механических систем, описывае-

мых дифференциальным, а не разностным уравнением. Общей задачей классической механики является определение координат и скоростей системы частиц, подверженных действию определенных сил. Например, в гл. 4 мы рассмотрели задачу двух тел в небесной механике и смогли предсказать движение в любой момент времени. Удивительным является то, что мы не в состоянии предсказать долговременное поведение для большинства орбит. Для многих механических систем характер движения можно определить приблизительно, комбинируя аналитические и численные методы. При определенных условиях эти системы проявляют хаотическое поведение, т.е. их траектории оказываются очень чувствительными к начальным данным и к любым приближениям, сделанным в ходе вычисления траектории. (Вспомните результаты задачи 7.2б для орбиты стандартного отображения в хаотическом режиме.) Классическая механика не так проста, как это описано в большинстве традиционных учебников!

Специальный случай нелинейной системы, который мы рассмотрим, представляет хорошо знакомый математический маятник (см. гл. 5). Чтобы движение такого маятника было более интересным, мы предположим, что маятник с затуханием и точка подвеса движется вертикально. Второй закон Ньютона для этой системы записывается (см. работы Маклафлина и Персиваля, Ричардса) в виде

$$\frac{d^2\theta}{dt^2} = -\gamma \frac{d\theta}{dt} - [\omega_0^2 + 2A\cos\omega t]\sin\theta, \tag{7.14}$$

где θ — угол, образуемый маятником с вертикальной осью, γ — коэффициент затухания, $\omega_0^2 = g/L$ — собственная частота осциллятора, а ω и A — частота и амплитуда внешней силы. Заметим, что эффект вертикального ускорения точки подвеса эквивалентен зависимости гравитационной силы от времени.

Конечно, в принципе (и в задаче 7.7) можно найти решения, описывающие динамику такой системы. Но как проанализировать результаты? В гл. 5 мы исследовали движение гармонического осциллятора с помощью фазовой плоскости, т.е. с помощью графиков скорости в зависимости от координаты. (Было бы неплохо дорешать задачи 5.2д, 5.3в и 5.5д, если вы их до сих пор не решили.) Если скорость постоянна, то траектория представляет собой горизонтальную прямую на фазовой плоскости. Возможно ли, чтобы траектория была вертикальной прямой на фазовой плоскости? Для простого гармонического движения траекто-

рией на фазовой плоскости является эллипс. Если имеется затухание, то траектория представляет собой скручивающуюся спираль. Какой будет траектория на фазовой плоскости мяча, падающего вертикально вниз в воздухе? График траектории на фазовой плоскости может принимать разнообразный вид. Если сила не зависит от времени, то, как показано (см. книгу Дж. Мариона), график траектории не может пересекать себя. Фазовые кривые, которые замкнуты, называются периодическими.

Какое движение ожидается в случае математического маятника с затуханием под действием вынуждающей силы? Поскольку маятник с затуханием, то предполагается, что в отсутствие внешней силы маятник возвратится в состояние покоя, т.е. точка $\{x = 0, \ v = 0\}$ является устойчивым аттрактором. Для больших значений A можно ожидать установившееся периодическое движение, т.е. предельный цикл с периодом внешней силы $T = 2\pi/\omega$. Проявляет ли система аналогичное поведение при еще больших значениях A? Поскольку нас главным образом интересуют неподвижные точки (устойчивые или неустойчивые), то простой способ анализа движения заключается в отмечании точки на фазовой плоскости в моменты времени, кратные периоду внешней силы. Таким образом, мы построим график $d\theta/dt$ в зависимости от θ в моменты времени, равные nT. Такой график в фазовом пространстве называется *отображением Пуанкаре*. Если у системы имеется период T, то отображение Пуанкаре состоит из единственной точки. Если период системы равен nT, то это отображение состоит из n точек.

В программе **nonlinear** строится график отображения Пуанкаре для математического маятника, описываемого уравнением (7.14). Значения θ и $d\theta/dt$ печатаются в моменты времени nT, а квадратик строится в точке $\{\theta, \ d\theta/dt\}$. Если система имеет период 1, т.е. если наносятся те же самые значения $\{\theta, \ d\theta/dt\}$ в моменты времени nT, то квадратик будет мерцать, указывая на то, что в этой точке уже есть квадратик. Клавиши 'i' и 'd' можно использовать для увеличения или уменьшения величины A на dA. Так как несколько первых значений $\{\theta, \ d\theta/dt\}$ соответствуют переходному процессу, может оказаться желательным очистить экран и начать построение графика нового отображения Пуанкаре, не меняя значения A, θ или $d\theta/dt$. Стирание экрана выполняется нажатием любых клавиш 'i', 'd' или 's'. Последняя клавиша останавливает выполнение программы.

```
PROGRAM nonlinear          ! построение графика отображения Пуанкаре
CALL initial(theta, ang_vel, gamma, A, n, dt, dA, r)
DO
  CALL Euler(theta, ang_vel, gamma, A, n, dt)
  CALL Poincare(theta, ang_vel, A, k, dA, r)
LOOP UNTIL k = ord("s")
END
SUB initial(theta, ang_vel, gamma, A, n, dt, dA, r)
 INPUT prompt "начальный угол = ": theta
 INPUT prompt "начальная угловая скорость = ": ang_vel
 INPUT prompt "постоянная затухания = ": gamma
 INPUT prompt "амплитуда внешней силы = ": A
 LET dt = 0.01             ! шаг по времени
 LET n = pi/dt             ! число итераций между точками графика
 LET t = 0                 ! начальное время
 LET dA = 0.01             ! приращение амплитуды внешней силы
 SET window -4,4,-4,4
 PRINT using "A = ***.*****": A
 PLOT LINES: -4,0; 4,0     ! построение осей
 PLOT LINES: 0,-4; 0,4
 LET r = 0.05              ! размеры квадратика
END SUB

SUB Euler(theta, ang_vel, gamma, A, n, dt)
! применение алгоритма Эйлера–Кромера
 FOR i = 1 to n
     LET accel = -gamma*ang_vel
     LET accel = accel - (1 + 2*A*cos(2*t))*sin(theta)
     LET ang_vel = ang_vel + accel*dt
     LET theta = theta + ang_vel*dt
     IF theta > pi then LET theta = theta - 2*pi
     IF theta < -pi then LET theta = theta + 2*pi
 NEXT i
END SUB
```

```
SUB Poincare(theta, ang_vel, A, k, da, r)
 ! наносит точки и изменяет при необходимости амплитуду внешней силы
 BOX CLEAR theta - r, theta + r, ang_vel - r, ang_vel + r
 BOX AREA theta - r, theta + r, ang_vel - r, ang_vel + r
 SET cursor 2,1
 PRINT using "theta = ***.*****": theta
 PRINT using "ang_vel = ***.*****": ang_vel
 IF key input then
      GET KEY k
      IF k = ord("i") then LET A = A + da
      IF k = ord("d") then LET A = A - da
      CLEAR
      PLOT LINES: -4,0; 4,0    ! построение новых осей
      PLOT LINES: 0,-4; 0,4
      PRINT using "A = ***.*****": A
 END IF
END SUB
```

***ЗАДАЧА 7.7. Движение математического маятника с затуханием и вынуждающей силой**

а. В программе **nonlinear** строится график отображения Пуанкаре для математического маятника с затуханием и под воздействием вынуждающей силы. Выбираются значения $\omega_0 = 1$ и $\omega = 2$, так чтобы период внешней силы T равнялся π. Для этих значений параметров удобно выбрать шаг $\Delta t = 0.3$. Используйте значения $\gamma = 0.2$ и $A = 0.85$ и вычислите период колебаний маятника. Проварьируйте начальные значения θ и $d\theta$? Зависит ли аттрактор от начальных условий? Помните о том, что надо проигнорировать переходный период.

б. Амплитуда A играет роль контрольного параметра движения системы. Найдите период и аттракторы для значений $A = 0.1$, 0.25, 0.5, 0.7, 0.75, 0.85, 0.95, 1.00, 1.02, 1.031, 1.033, 1.036 и 1.05. Заметим, что в отличие от квадратичного отображения период равен 2π для $A < 0.71$, π для $0.72 < A < 0.79$ и 2π в остальных случаях.

в. Для понимания связи между отображением Пуанкаре и графиком зависимости θ от времени измените программу **nonlinear** так, чтобы

θ и $d\theta/dt$ наносились на график на каждом временно́м шаге, а не только в моменты времени $n\pi$. Опишите поведение маятника в каждом из случаев, рассмотренных в п. «б».

г. Первое удвоение периода происходит при значении $A \approx 0.79$. Найдите значения A, соответствующие последующим удвоениям периода, и воспользуйтесь этими значениями для определения показателя δ, определяемого по формуле (7.13). Сравните свой результат δ со значением, соответствующим одномерному квадратичному отображению. Совпадает ли полученное вами значение с найденным для квадратичного отображения?

д. Повторите вычисления п. п. «б»–«г» для значения $\gamma = 0.05$. Какие выводы можно сделать об эффекте затухания? [Анализ этой системы можно найти в статье Маклафлина (1981).]

7.6. ДВУМЕРНОЕ ОТОБРАЖЕНИЕ

В настоящий момент вы, вероятно, заинтересовались либо нелинейными системами и уже прочитали некоторые статьи из списка литературы, либо готовитесь перейти к чтению гл. 8 и снова рассмотреть линейные системы. Тем не менее мы не можем воздержаться от рассмотрения двумерных систем в задаче 7.8, проявляющих «странное» поведение.

*ЗАДАЧА 7.8. Двумерное отображение

Рассмотрите последовательность точек $\{x_n, y_n\}$, генерируемых двумерным отображением:

$$x_{n+1} = y_n + 1 - ax_n^2 \tag{7.15а}$$

$$y_{n+1} = bx_n. \tag{7.15б}$$

Отображение (7.15) предложено Хеноном (см. список литературы), который установил, что (7.15) ведет себя аналогично системе дифференциальных уравнений, предложенной для изучения крупномасштабных климатических моделей.

а. Проитерируйте (7.15) для значений $a = 1.4$ и $b = 0.3$ и нанесите на график первые 10000 точек, стартующих из начальной точки

$\{x_0 = 0, \quad y_0 = 0\}$. Убедитесь в том, что ваши вычисления нового значения y используют только старое значение x. Задайте инструкцию **SET window** так, чтобы изображались все значения точек орбиты внутри рамки, которая строится с помощью инструкции **BOX LINES** –1.5, 1.5, –0.45, 0.45. Постройте аналогичный график для начального значения $\{x_0 = 0.63135448, \quad y_0 = 0.18940634\}$. Сравните полученные две кривые. Зависит ли форма этих кривых от начальных условий?

б. Увеличьте шкалу графика так, чтобы все точки с рамкой, построенной с помощью инструкции **BOX LINES** 0.50, 0.75, 0.15, 0.21, были показаны на экране. Постройте новый график для второго начального значения и увеличьте число вычисляемых точек до 10^5. Затем постройте график, помещающий все точки внутрь рамки, построенной с помощью **BOX LINES** 0.6305, 0.6325, 0.1889, 0.1895. Как ведут себя кривые внутри каждого окна? Должны ли кривые иметь одинаковую структуру? (Необходимо будет увеличить число точек до 10^6.) Указывают ли ваши результаты на то, что процесс размножения «кривых» будет продолжатся до бесконечности и что каждая «кривая» образуется из бесконечного числа квазипараллельных кривых? Существует ли область на плоскости, которую точки не могут покинуть? Эта область известна как *аттрактор Хенона*, являющийся примером *странного аттрактора*.

ЛИТЕРАТУРА

Ralph H. Abraham, Cristopher D. Shaw, Dynamics—The Geometry of Behavior, Vol. 1—4, Aerial Press, Santa Cruz, CA, 1984. В этих четырех томах и в монографии Шоу (см. ниже) авторы объясняют поведение сложных динамических систем, используя многочисленные наглядные образы.

Hao Bai-Lin, Chaos, World Scientific, 1984. Сборник статей Фейгенбаума, Хенона, Мэя, Теста и др., на которые мы ссылаемся в этой главе. Представляют интерес также статьи, посвященные экспериментальному изучению хаотических явлений.

James P. Crutchfield, J. Doyne Farmer, Norman H. Packhard, Robert S. Shaw, Chaos, Sci. Amer. **255**, 6, 46—57 (December, 1986).

J. P. Crutchfield, J. D. Farmer, B.A. Huberman, Fluctuations and Simple Chaotic Dynamic, Phys. Repts., **92**, 45 (1982).

Predrag Cvitanovic, Universality in Chaos, Adam Hilger, 1984. Сборник перепечаток некоторых статей, на которые ссылаются в сборнике под ред. Бай-Лина. Представляют интерес также перепечатки статей, касающихся приложений к химическим, оптическим и биологическим системам.

Robert Devaney, Introduction to Chaotic Dynamical Systems, Benjamin/Cummings, 1986.

J. P. Eckmann, Roads to turbulence in dissipative dynamical system, Rev. Mod. Phys. **53**, 643 (1981).

M. Henon, A two-dimensional mapping with a strange attractor, Comm. Math. Phys. **50**, 50 (1976). Перепечатано в сборниках под ред. Бай-Лина и Цвитановича. [Имеется перевод: Хенон М., Двумерное отображение со странным аттрактором, в сб. Странные аттракторы. — М., Мир, 1981.]

T. A. Heppenheimer, Routes to chaos, Mosaic **17**, No.2, pg.2 (Summer, 1986).

Douglas R. Hofstadter, Metamagical Themas, Sci. Amer. **245**, 22–43 (November, 1981). Расширенный вариант этой статьи в работе *Douglas R. Hofstadter*, Metamagical Themas, Basic Books, 1985.

Leo P. Kadanoff, Roads to chaos, Physics Today **36**, 46 (December, 1983).

Jerry B. Marion, Classical Dynamics of Particles and Systems, 2nd ed., Academic Press, 1970. В гл. 5 обсуждаются графики в фазовом пространстве и нелинейные уравнения с различными приложениями.

Robert M. May, Simple Mathematical Models with Very Complicated Dynamics, Nature **261**, 459 (1976). Перепечатано в сборниках под ред. Бай-Лина и Цвитановича.

J. B. McLaughlin, Period-doubling bifurcations and chaotic motion for a parametrically forced pendulum, J. Stat. Phys. **24**, 375 (1981).

E. Ott, Strange attractors and chaotic motions of dynamical systems, Rev. Mod. Phys. **53**, 655 (1981).

Ian Percivel, Derec Richards, Introduction to Dynamics, Cambridge University Press, 1982. Современная и написанная на высоком уровне книга, в которой вводятся многие понятия теории устойчивости и фазовых кривых. Рассмотренный в разд. 7.4 вывод гамильтониана для вынужденных колебаний маятника с затуханием, воспроизведен в данной книге в примере 5.7 гл. 5.

Robert Shaw, The Dripping Faucet as a Model Chaotic System, Aerial Press, Santa Cruz, CA, 1984.

James Testa, Jose Perez, Carson Jeffries, Evidence for universal chaotic behavior of driven nonlinear oscillator, Phys. Rev. Lett. **48**, 714 (1982). (Перепечатано в сборниках под ред. Бай-Лина и Цвитановича.)

N. B. Tufillaro, A.M. Albano, Chaotic dynamics of a bouncing ball, Amer. J. Phys. **54**, 939 (1986). Авторы описывают на высоком уровне эксперимент с мячом, отскакивающим от вибрирующего стола.

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

Шустер Г., Детерминированный хаос: введение. — М.: Мир, 1988. В гл. 3 подробно исследуются поведение квадратичных отображений, бифуркации и вычисляются α и δ. В гл. 6 рассматривается отображение Хенона. Рассчитана на неподготовленного читателя.

Николис Дж., Динамика иерархических систем: эволюционное представление. — М.: Мир, 1989. Книга написана на том же уровне, что и предыдущая. В гл. 6 подробно рассмотрены вопросы поведения нелинейных отображений, бифуркации и переход к хаосу.

Фейгенбаум М., Универсальное поведение нелинейных систем, УФН, т. 141, вып. 3, 343—374 (1983). Перевод одной из основополагающих работ создателя этого научного направления.

Странные аттракторы, Новое в зарубежной науке, Математика, 22. — М.: Мир, 1981. Сборник основополагающих работ различных авторов, рассчитанный на подготовленного читателя. В частности, имеется перевод статьи Хенона (см. список литературы). Сборник рассчитан на подготовленного читателя.

ПРИЛОЖЕНИЕ 7А. УСТОЙЧИВОСТЬ НЕПОДВИЖНЫХ ТОЧЕК

Получим аналитические выражения для неподвижных точек стандартного квадратичного отображения, задаваемого выражением (7.6). Условие неподвижности записывается следующим образом:

$$x^* = f(x^*). \tag{7.16}$$

Используя (7.6), получим две неподвижные точки:

$$x^* = 0 \qquad \text{и} \qquad x^* = 1 - \frac{1}{4r}. \tag{7.17}$$

Поскольку x должно оставаться положительным, то единственной неподвижной точкой для значения $r < 1/4$ является $x = 0$. Для определения устойчивости точки x^* положим

$$x_n = x^* + \Delta_n \tag{7.18а}$$

и

$$x_{n+1} = x^* + \Delta_{n+1}. \tag{7.18б}$$

Тогда, поскольку $|\Delta_n| \ll 1$, получим

$$x_{n+1} = f(x^* + \Delta_n) \approx f(x^*) + \Delta_n f'(x^*)$$

$$= x^* + \Delta_n f'(x^*). \tag{7.19}$$

Тогда, используя (7.18б) и (7.19), получим

$$\Delta_{n+1}/\Delta_n = f'(x^*). \tag{7.20}$$

Если $|f'(x^*)| > 1$, то траектория будет удаляться от x^*, поскольку $|\Delta_{n+1}| > |\Delta_n|$. Противоположное утверждение верно для случая $|f'(x^*)| < 1$. Таким образом, критерий локальной устойчивости для неподвижной точки x^* формулируется следующим образом

1. $|f'(x^*)| < 1$ — x^* является устойчивой;
2. $|f'(x^*)| = 1$ — x^* является условно устойчивой;
3. $|f'(x^*)| > 1$ — x^* является неустойчивой.

Если x^* условно устойчива, тогда необходимо рассматривать вторую производную $f''(x)$ и итерационное приближение x^* с отклонением от x^* обратно пропорциональным корню квадратному из числа итераций. Для $f(x)$, задаваемой выражением (7.6), производные в неподвижных точках равны соответственно

$$f'(0) = \frac{d}{dx}[4rx(1-x)]\Big|_{x=0} = 4r \tag{7.21}$$

и

$$f'(x^*) = \frac{d}{dx}[4rx(1-x)]\Big|_{x=1-1/4r} = 2 - 4r. \qquad (7.22)$$

Выражение (7.22) можно использовать для нахождения интервала значений r, в котором точки $x^* = 0$ и $x^* = 1 - \frac{1}{4r}$ являются устойчивыми.

ВОЛНОВЫЕ ЯВЛЕНИЯ

8

В данной главе моделируется линейная цепочка связанных осцилляторов, выделяются свойства цепочки, относящиеся к волновым явлениям, и в пределе непрерывной цепочки выводится линейное волновое уравнение. Демонстрируются такие волновые явления, как интерференция, дифракция, рефракция и поляризация. Рассматриваются также ряд Фурье и принцип Ферма.

8.1. ВВЕДЕНИЕ

Для описания как волнового, так и колебательного движений мы пользуемся такими понятиями, как период, амплитуда и частота. Чтобы понять, как связаны между собой оба явления, рассмотрим однородный гибкий шнур, который натянут и один его конец закреплен (рис. 8.1).

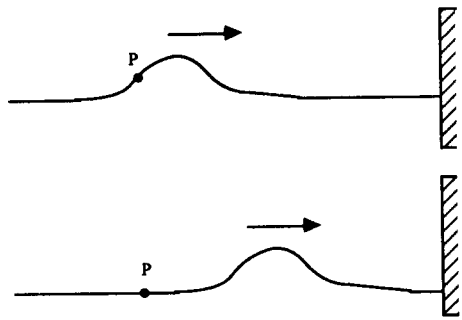

Рис. 8.1. Волновой импульс, распространяющийся по натянутому шнуру. Движение участка P локализовано и происходит перпендикулярно движению волны.

Если мы встряхнем шнур за другой конец, то по нему будет *распространяться* волновой импульс со скоростью, определяемой натяжением и инерционными свойствами шнура. В процессе распространения волнового импульса вдоль шнура каждый участок шнура движется вверх и вниз перпендикулярно направлению импульса. Мы знаем, что на *макроскопическом* уровне наблюдается поперечная волна, движущаяся вдоль шнура, а движение отдельных участков шнура не рассматривается. В противоположность этому на *микроскопическом* уровне мы видим дискретные частицы, испытывающие колебательное движение в направлении, перпендикулярном движению волны.

Мы начнем с микроскопической картины и рассмотрим в разд. 8.2 колебательное движение линейной цепочки частиц, соединенных пружинками. При достаточно большом числе частиц в цепочке движение продольной волны можно моделировать, численно решая уравнения движения Ньютона для отдельных частиц. Мы обнаружим, например, что энергия передается вдоль цепочки, хотя каждый осциллятор остается вблизи своего положения равновесия. Мы увидим также, что самое общее движение системы N частиц можно представить как суперпозицию N незави-

симых простых гармонических движений. Это представление имеет более широкое применение, и в разд. 8.3 мы познакомимся с представлением произвольной периодической функции рядом Фурье. В разд. 8.4, получая волновое уравнение для механических волн, мы перейдем от микроскопического к макроскопическому описанию линейной цепочки. Кроме того, исследуются свойства решений волнового уравнения.

Остальная часть этой главы посвящена анализу волнового движения. В разд. 8.5 мы рассматриваем явления интерференции и дифракции, а в разд. 8.6 иллюстрируем различные виды поляризованных волн. В разд. 8.7 мы применяем принцип наименьшего времени Ферма к задачам геометрической оптики.

8.2. СВЯЗАННЫЕ ОСЦИЛЛЯТОРЫ

В гл. 6 мы моделировали динамику системы N взаимодействующих частиц, но не рассматривали их коллективное движение, например распространение звуковых волн. Теперь рассмотрим гораздо более простую систему N частиц, на которой можно изучать распространение волн. Нашей главной целью будет познакомиться с понятиями нормальных колебаний, биений и распространения энергии.

Рассмотрим одномерную цепочку из N частиц, каждая массой m с одинаковым равновесным расстоянием a. Частицы связаны пружинками нулевой массы с силовой постоянной k_c, за исключением двух крайних пружинок, силовая постоянная которых равна k (рис. 8.2). Пусть u_i

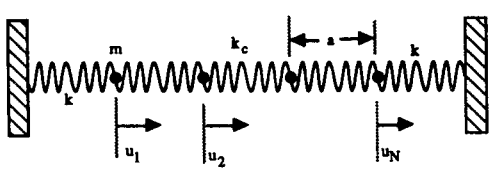

Рис. 8.2. Продольные колебания точечных масс, соединенных пружинками.

есть смещение i-й массы из равновесного положения вдоль оси системы. Концы левой и правой пружинок считаем неподвижными. Неподвижность концевых точек выразим условием

$$u_0 = u_{N+1} = 0. \tag{8.1}$$

Поскольку сила, действующая на каждую отдельную массу, определяется только сжатием или растяжением соединенных с ней пружинок, уравнение движения i-й частицы имеет вид

$$m\frac{d^2u_i}{dt^2} = -k_c(u_i - u_{i+1}) - k_c(u_i - u_{i-1}) =$$

$$= k_c(2u_i - u_{i+1} - u_{i-1}), \qquad i = 2,\ldots,N-1. \tag{8.2}$$

Уравнения для частиц $i = 1$ и N, ближайших к стенкам, имеют вид

$$m\frac{d^2u_1}{dt^2} = -k_c(u_1 - u_2) - ku_1, \tag{8.3а}$$

$$m\frac{d^2u_N}{dt^2} = -k_c(u_N - u_{N-1}) - ku_N. \tag{8.3б}$$

Заметим, что при $k_c = 0$ указанная система уравнений для u_i распадается на независимые уравнения и движение каждой точечной массы не зависит от ее соседей. Представленные уравнения движения описывают *продольные* колебания, т.е. движение вдоль системы. Можно непосредственно показать, что уравнения того же вида справедливы и для *поперечных* колебаний N одинаковых точечных масс, расположенных на растянутой пружинке нулевой массы на одинаковом расстоянии друг от друга (см. книгу А. Френча).

Для моделирования динамического поведения N связанных масс воспользуемся для расчета смещений и скоростей частиц алгоритмом Эйлера—Кромера. Программа **oscillators** рисует смещение как функцию времени не более чем для четырех частиц. Мы рассматриваем случай, когда все частицы имеют одинаковую массу, которую мы полагаем равной единице.

```
PROGRAM oscillators    ! вычисляется смещение N связанных осцилляторов
DIM u(0 to 21), vel(20)
CALL initial(N, u, vel, kc, k, dt, tmax)
CALL screen(N, nplot, tmax, dy)
CALL move(N, u, vel, kc, k, dt, nplot, tmax, dy)
END
```

```
SUB initial(N, u(), vel(), kc, k, dt, tmax)
  INPUT prompt "число частиц = ":N
  INPUT prompt "шаг по времени = ":dt
  INPUT prompt "продолжительность = ":tmax
  INPUT prompt "силовая постоянная kc = ": kc ! внутренние пружинки
  LET k = 1                                   ! пристеночные пружинки
  DATA 0.5, 0, 0                              ! данные для N = 2
  FOR i = 1 to N                              ! начальные условия
    LET vel(i) = 0
    READ u(i)
  NEXT i
END SUB

SUB screen(N, nplot, tmax, dy)
  LET dy = 2              ! расстояние между графиками на экране
  SET WINDOW -1, tmax + 1, -dy, 3*dy
  LET ntick = 100
  LET dx = tmax/ntick     ! расстояние между делениями
  LET Ly = 0.1*dx         ! "высота" делений
  LET nplot = min(4, N)   ! число осцилляторов для рисования графиков
  LET row = -1
  FOR iplot = 1 to nplot
    PLOT LINES: 0, row; tmax, row    ! отмечаем равновесное расстояние
    FOR itick = 1 to ntick           ! рисуем деления оси
      LET col = itick*dx
      PLOT LINES: col, row; col, row + Ly      ! рисуем деления
    NEXT itick
    LET row = row + dy
  NEXT iplot
END SUB

SUB move(N, u(), vel(), kc, k, dt, nplot, tmax, dy)
  DIM a(20)
  DO
   LET t = t + dt          ! время
   FOR i = 2 to (N - 1)    ! ускорение точек, не связанных со стенками
     LET a(i) = kc*(u(i+1) + u(i-1) - 2*u(i))
   NEXT i
   LET a(1) = kc*(u(2) - u(1)) - k*u(1)      ! ускорения концевых масс
   LET a(N) = kc*(u(N-1) - u(N)) - k*u(N)
```

```
    FOR i = 1 to N          ! алгоритм Эйлера–Кромера
      LET vel(i) = vel(i) + a(i)*dt
      LET u(i) = u(i) + vel(i)*dt
    NEXT i
    LET row = -1
    FOR iplot = 1 to nplot
      PLOT POINTS: t,u(iplot) + row
      LET row = row + dy
    NEXT iplot
  LOOP until t > tmax
END SUB
```

В следующих трех задачах исследуется влияние суперпозиции гармонических колебаний с разными частотами. Мы увидим, что в общем случае могут возникать сложные структуры и что их можно описать как суперпозицию *нормальных* колебаний.

ЗАДАЧА 8.1. Движение двух связанных осцилляторов

а. Используйте программу **oscillators** с $N = 2$ (число частиц). Массивы $u(i)$, $vel(i)$ и $a(i)$ содержат смещение, скорость и ускорение i-й частицы. Во всех вариантах этой задачи начальные скорости обеих частиц задавайте равными нулю. Выберите начальные условия $u_0(1) = 0.5$, $u_0(2) = 0$ и вычислите зависимость от времени $u(1)$ и $u(2)$ для $\{k = 1, \ k_c = 0.8\}$ и $\{k = 1, \ k_c = 1\}$. Определите подходящие значения шага по времени dt и времени моделирования $tmax$. (Напомним, что масса частиц принята равной единице.) Опишите качественное поведение $u(1)$ и $u(2)$ в каждом случае. Можно ли определить период колебаний в первом случае? Чему равен период колебаний для второй пары значений k?

б. Задайте $k = 1$ и $k_c = 0.2$. Поскольку $k_c < k$, можно считать, что пружинки *слабо связаны*. Получите временную зависимость смещения первой частицы. Удается ли вам распознать два вида колебаний, наложенных друг на друга? Чему равен период колебаний амплитуды? Чему равен промежуток времени между нулями смещения? Вычислите соответствующую угловую частоту каждого колебания. Как смещение второй частицы соотносится со смещением первой частицы? Определите качественно, как изменятся частоты каждого колебания при $k_c = 0.1$.

в. Выберите начальные условия $u_0(1) = u_0(2) = 0.5$, т.е. обе частицы имеют одинаковые начальные смещения. Задайте $k_c = 0.1$ и $k = 1$ и опишите наблюдаемое движение. Вычислите энергию (кинетическую плюс потенциальную) каждой частицы как функцию времени и опишите ее качественное поведение. Зависит ли период колебания от k_c? Как зависит этот период от k?

г. Рассмотрите начальные условия $u_0(1) = -u_0(2) = 0.5$, т.е. начальные смещения обеих частиц равны по величине, но направлены в противоположные стороны. Имеется ли в этом случае простое гармоническое колебание? Вычислите период T_1 для $\{k = 1, k_c = 1\}$, $\{k = 2, k_c = 1\}$ и $\{k = 1, k_c = 2\}$. Проанализируйте полученные результаты для ω_1^2 ($\omega_1 = 2\pi/T_1$) и определите зависимость ω_1^2 от k и k_c. Как ведет себя энергия каждой частицы в зависимости от времени?

ЗАДАЧА 8.2. Отклик на внешнюю силу

Приложите к частице 1 внешнюю вынуждающую силу $F(t)$, такую чтобы $F(t) = F_0 \cos \omega t$. Нарисуйте график функции $u_1(t)$ и определите ее максимальную установившуюся амплитуду $A(\omega)$ для каждого значения ω. Подтвердите, что вблизи резонанса $A(\omega)$ проявляет быстрый рост в зависимости от ω, а на резонансной частоте величина $u_1(t)$ неограниченно возрастает. (Напомним, что мы имеем дело с системой без затухания.) Определите резонансные частоты для пар значений k, которые рассматривались в п.п. «б» — «г» задачи 8.1. Насколько эти значения ω отличаются от найденных в задаче 8.1?

ЗАДАЧА 8.3. Суперпозиция колебаний

Результаты задач 8.1 и 8.2 наводят на мысль, что всякое произвольное движение рассмотренной системы можно представить в виде

$$u(1) = A_1 \cos(\omega_1 t + \delta_1) + A_2 \cos(\omega_2 t + \delta_2), \qquad (8.4а)$$

$$u(2) = A_1 \cos(\omega_1 t + \delta_1) - A_2 \cos(\omega_2 t + \delta_2). \qquad (8.4б)$$

Постоянные A_1, A_2, δ_1 и δ_2 можно выразить через начальные значения смещения и скорости каждой частицы. Определите эти постоянные для $u_0(1) = 0.5$, $u_0(2) = 0$ и $v_0(1) = v_0(2) = 0$. Удостоверь-

тесь в том, что движение, предсказываемое формулами (8.4), согласуется с измеренными вами значениями $u(1)$ и $u(2)$ при $k = 1$ и $k_c = 0.8$, которые вычислялись в п. «а». Чему равны периоды $u(1)$ и $u(2)$?

Посмотрим еще раз, что же мы установили в результате решения задачи 8.1 о двух связанных частицах. Действие внутренней пружинки, имеющей силовую постоянную k_c, связывает движение обеих частиц, и они уже не движутся независимо. При специальных начальных условиях проявляется только одна частота колебаний. Получающееся при этом движение называется *нормальным колебанием*, а соответствующие частоты называются *нормальными частотами*. Верхняя частота равна $\omega_1 = [(k + 2k_c)/m]^{1/2}$. В этом режиме обе частицы колеблются точно в противофазе; иначе говоря, их смещения направлены в противоположные стороны. Движение при нижней частоте $\omega_2 = (k/m)^{1/2}$ соответствует колебаниям обеих частиц точно в фазе. Почему ω_2 не зависит от k_c?

Общее движение рассмотренной двухчастичной системы представляет собой суперпозицию обоих нормальных колебаний. Если ω_1 и ω_2 не связаны простым соотношением, результирующее смещение является сложной функцией времени. Однако если связь слабая, ω_1 и ω_2 почти равны и в $u(1)$ и $u(2)$ обнаруживаются *биения*. В этом случае частицы быстро колеблются с угловой частотой $\frac{1}{2}(\omega_1 + \omega_2)$ и амплитудой, меняющейся по синосоидальному закону с частотой биений $\frac{1}{2}(\omega_1 - \omega_2)$. Явление биений широко распространено в природе. Наверное, вам доводилось слышать два источника звука со слегка различными частотами, например звучание двух расстроенных скрипок, играющих одну и ту же ноту. Результат воспринимается как один звук переменной силы.

Мы нашли также, что при действии на систему внешней силы, приложенной к любой частице (или обеим частицам), система входит в резонанс, если частота приложенной силы равна любой из нормальных частот. В задачах 8.4 и 8.5 мы воспользуемся этим методом для определения частот нормальных колебаний.

ЗАДАЧА 8.4. Три связанных осциллятора

а. Выполните программу **oscillators** с $N = 3$, $k_c = 0.2$, $k = 1$ и произвольными, но не нулевыми начальными смещениями. Опишите зависимость смещения частиц от времени.

б. Рассмотрите следующие начальные условия, приведенные в табл. 8.1. (Все начальные скорости равны нулю.) Если эти начальные условия отвечают нормальным колебаниям, определите соответствующие нормальные частоты.

ТАБЛИЦА 8.1. Начальные условия, рассматриваемые в задаче 8.4б

	$u_0(1)$	$u_0(2)$	$u_0(3)$
Случай 1	0.5	0.5	0.5
Случай 2	0.5	−0.5	0.5
Случай 3	0.5	0	−0.5

в. Приложите к частице 1 внешнюю вынуждающую силу и определите частоты нормальных колебаний. Сравните полученные результаты с частотами, полученными в п. «б». Сколько всего имеется нормальных колебаний?

*ЗАДАЧА 8.5. N связанных осцилляторов

а. Возьмите $k_c = k = 1$ и $N = 10$. (Оптимальный выбор N зависит от быстродействия вашего компьютера и вашего терпения.) Найдите все нормальные колебания, прикладывая внешнюю силу к одной из частиц и определяя резонансные частоты. Возбуждайте систему в течение по крайней мере нескольких периодов внешней силы и вычисляйте установившуюся амплитуду смещения каждой частицы для каждого значения ω. Пройдитесь по значениям ω в диапазоне от $0.2(k/m)^{1/2}$ до $3(k/m)^{1/2}$. Если вы сочтете, что находитесь вблизи резонанса, то для получения более точной оценки используйте несколько дополнительных значений ω. Сколько всего имеется нормальных колебаний? Более систематический метод определения нормальных колебаний предложен Уильямсом и Марисом (см. список литературы), которые применили свой метод к неупорядоченным системам со случайными массами.

б. Сравните результаты, полученные в п. «а», с аналитическим результатом

$$\omega_n^2 = \frac{4k}{m} \sin^2 \frac{n\pi}{2(N+1)}, \tag{8.5}$$

где N — число частиц, а n — номер колебания, $n = 1, 2, \ldots, N$.

Еще одним важным свойством системы связанных осцилляторов является передача энергии. В задаче 8.6 мы создаем возмущение цепочки на одном конце и определяем время, за которое это возмущение проходит заданное расстояние.

ЗАДАЧА 8.6. Скорость распространения в линейной цепочке

а. Рассмотрите линейную цепочку связанных осцилляторов с $k_c = k$, находящихся в состоянии покоя. Создайте возмущение, сообщив частице 1 начальное смещение $u_0(1) = 1$. Определите, через какое время для частиц $N/2$ и N в первый раз выполняются условия $|u(N/2)| \geq d$ и $|u(N)| \geq d$. Примите для своих первых запусков $N = 10$, $k = 1$ и $d = 0.3$. С помощью полученных результатов оцените v — скорость распространения возмущения. Чтобы убедиться в независимости своей оценки v от N, повторите расчеты для бо́льших значений N. (Достаточно получить для v качественные результаты.)

б. Как вы думаете, зависимость скорости распространения от силовой постоянной k будет возрастающей или убывающей функцией? Проведите моделирование и оцените v для различных значений k.

в. Создайте возмущение, прикладывая к частице 1 внешнюю силу $F(t) = \cos \omega t$. Оцените скорость распространения возмущения так же, как делали это в п. «а». Задайте $k = 1$ и рассмотрите значения $\omega = 0.1$ и 1. Объясните, почему скорость распространения зависит от ω. Может ли возмущение распространяться для $\omega = 4$? Каким образом система действует как механический фильтр? Объясните «фильтрующее» свойство системы на языке частоты нормальных колебаний.

*г. Рассмотрите «неупорядочную» систему при помощи приписывания частицам различных масс. Как влияет беспорядок на скорость распространения? (Обсуждение родственных научных проблем см. в статье Уильямса и Мариса.)

8.3. ФУРЬЕ-АНАЛИЗ

В задаче 8.3 мы установили, что смещение частицы можно представить в виде линейной комбинации нормальных колебаний, т.е. линейной суперпозиции синусоидальных членов. Такое разложение движения по различным частотам носит более общий характер. Можно показать, что, вообще говоря, произвольная периодическая функция $f(t)$ с периодом T может быть записана в виде ряда Фурье по синусам и косинусам

$$f(t) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty}(a_n \cos n\omega_0 t + b_n \sin n\omega_0 t), \tag{8.6}$$

где ω_0 — основная угловая частота

$$\omega_0 = \frac{2\pi}{T}. \tag{8.7}$$

В сумме (8.6) члены с синусом и косинусом, отвечающие $n = 2, 3, \ldots$, представляют собой вторую, третью и т.д. гармоники. *Коэффициенты Фурье* a_n и b_n выражаются формулами

$$a_n = \frac{2}{T}\int_{-T/2}^{T/2} f(t) \cos n\omega_0 t \, dt, \tag{8.8а}$$

$$b_n = \frac{2}{T}\int_{-T/2}^{T/2} f(t) \sin n\omega_0 t \, dt. \tag{8.8б}$$

Постоянный член $\frac{1}{2}a_0$ равен среднему значению $f(t)$.

Для точного представления произвольной функции $f(t)$ необходимо, вообще говоря, бесконечное число членов. На практике же обычно удается получить хорошее приближение к $f(t)$, учитывая относительно небольшое число членов. Приведенная ниже программа **Fourier** выводит на экран график суммы (8.6) для различных n и помогает нам наглядно представить точность, которую обеспечивают конечные суммы гармонических членов.

```
PROGRAM Fourier
DIM a(0 to 100), b(100)
CALL initial(N)
CALL screen(xmin, xmax, ymin, ymax, title$)
CALL coefficients(N, a, b)
CALL plot(N, a, b)
END

SUB initial(N)
  SET CURSOR 1,10
  INPUT prompt "введите число гармоник = ": N
END SUB

SUB screen(xmin, xmax, ymin, ymax, title$)
  LET xmin = 0
  LET xmax = 2*pi
  LET ymin = -2
  LET ymax = 2
  LET title$ = "Преобразование Фурье"
  CALL plot_axis(xmin, xmax, ymin, ymax, title$)      ! см. листинг в гл. 2
END SUB

SUB coefficients(N, a(), b()) ! выдает коэффициенты Фурье для частного случая
  LET a(0) = 0
  FOR imode = 1 to N
    LET a(imode) = 0
    IF mod(imode, 2) <> 0 then
      LET b(imode) = 2/(pi*imode)
    ELSE
      LET b(imode) = 0
    END IF
  NEXT imode
END SUB

SUB plot(N, a(), b())  ! вычисляется ряд Фурье и рисуется график функции
  LET npoint = 100
  LET dx = 2*pi/npoint
  FOR ipoint = 1 to npoint
    LET x = dx*ipoint
    LET f = a(0)/2
```

```
    FOR imode = 1 to N
      IF a(imode) <> 0 then LET f = f + a(imode)*cos(imode*x)
      IF b(imode) <> 0 then LET f = f + b(imode)*sin(imode*x)
    NEXT imode
    PLOT LINES: x,f;
  NEXT ipoint
END SUB
```

ЗАДАЧА 8.7. Фурье-анализ

а. С помощью программы **Fourier** посмотрите, как можно представить негармоническую (но периодическую) функцию суммой гармонических функций. Рассмотрите следующую сумму, представленную в подпрограмме **coefficients**:

$$f(t) = \frac{2}{\pi}\left(\sin t + \frac{1}{3}\sin 3t + \frac{1}{5}\sin 5t + \ldots\right). \qquad (8.9)$$

Как выглядит $f(t)$, если в (8.9) сохранить только первые три члена? Увеличивайте число членов до тех пор, пока вы не убедитесь, что формула (8.9) представляет требуемую функцию достаточно точно. Какая функция представляется данной бесконечной суммой?

б. Используя формулы (8.8), покажите, что коэффициенты Фурье «пилообразной» функции, изображенной на рис. 8.3, равны $a_n = 0$, $b_n = (2/n\pi)(-1)^{n-1}$ для $n = 1, 2, 3, \ldots$. Модифицируйте программу **Fourier** для вывода графика суммы (8.6) с указанными коэффициентами и подтвердите свои результаты.

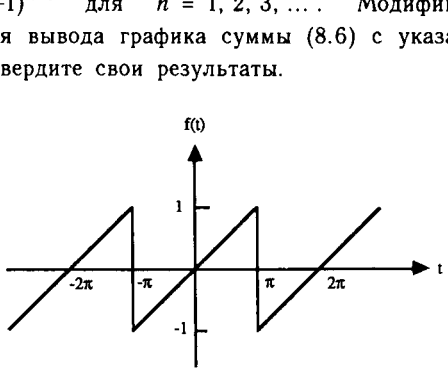

Рис. 8.3. «Пилообразная» функция, изучаемая в задаче 8.7б.

в. Какая функция представляется суммой (8.6) с коэффициентами $a_0 = 0$, $a_n = b_n = 1/n^2$ для $n \neq 0$?

8.4. ВОЛНОВОЕ ДВИЖЕНИЕ

До сих пор мы делали упор на аналогию между колебательными явлениями и волновым движением. В частности, мы нашли, что для системы N связанных осцилляторов колебания отдельных частиц приводят к распространению энергии на произвольные расстояния. Вся разница между волнами и колебательным движением связанных систем заключается в различии масштабов. Чтобы понять переход от колебательных явлений к волновым, рассмотрим опять продольные колебания линейной цепочки из N частиц с одинаковыми силовыми постоянными пружинок k и равным равновесным расстоянием a. Уравнения движения можно записать в виде [см. уравнение (8.2)]

$$\frac{d^2u_i}{dt^2} = -\frac{k}{m}(2u_i - u_{i+1} - u_{i-1}), \qquad i = 1,\ldots,N. \tag{8.10}$$

Рассмотрим предельный переход

$$N \longrightarrow \infty, \qquad a \longrightarrow 0 \tag{8.11}$$

при постоянной длине цепочки. Основной результат состоит в том, что в пределе (8.11) дискретные уравнения движения (8.10) можно заменить непрерывным *волновым уравнением*

$$\frac{\partial^2 u(x,t)}{\partial t^2} = \frac{1}{v^2}\frac{\partial^2 u(x,t)}{\partial x^2}, \tag{8.12}$$

где v имеет размерность скорости.

Волновое уравнение (8.12) можно получить следующим образом. Заменим $u_i(t)$, где i—*дискретная* переменная, на функцию $u(x,t)$, где x—*непрерывная* переменная. Затем перепишем уравнения (8.10) в виде

$$\frac{\partial^2 u(x,t)}{\partial t^2} = \frac{ka^2}{m}\frac{1}{a^2}[u(x+a,t) - 2u(x,t) + u(x-a,t)]. \tag{8.13}$$

Мы записали производную по времени как частную производную, так как

функция u зависит от двух переменных. Поскольку частицы распределены непрерывно, можно ввести величины $\mu = m/a$ и $T = k/a$. Если воспользоваться разложением в ряд Тейлора

$$u(x \pm a) = u(x) \pm a\frac{\partial u}{\partial x} + (a^2/2)\frac{\partial^2 u}{\partial x^2} + \ldots, \tag{8.14}$$

то легко показать, что при $a \longrightarrow 0$

$$\frac{1}{a^2}\left[u(x+a,t) - 2u(x,t) + u(x-a,t)\right] \longrightarrow \frac{\partial^2 u(x,t)}{\partial x^2}. \tag{8.15}$$

Подстановка (8.15) в (8.13) и дает волновое уравнение (8.12) с $v^2 = T/\mu$ (T — натяжение, μ — линейная плотность массы).

Волновое уравнение имеет множество решений. Так, например, решениями являются функции

$$u(x,t) = A\cos\frac{2\pi}{\lambda}(x \pm vt), \tag{8.16а}$$

$$u(x,t) = A\sin\frac{2\pi}{\lambda}(x \pm vt). \tag{8.16б}$$

Легко непосредственно убедиться в том, что любая функция вида $f(x - vt)$ или $f(x + vt)$ является решением уравнения (8.12). Поскольку волновое уравнение линейное и, следовательно, удовлетворяет принципу суперпозиции, то поведение волны произвольной формы можно понять, представляя по теореме Фурье ее форму в виде суммы синусоидальных волн. Поэтому в дальнейшем нам нужно рассмотреть только гармонические (синусоидальные) решения волнового уравнения.

Для более подробного изучения свойств решений (8.12) нам понадобится программа **waves**.

```
PROGRAM waves
CALL initial( A, v, lambda, dt)
CALL screen( A, lambda, dt, xmax, ntime, space)
CALL wavemotion( A, v, xmax, ntime, dt, space)
END
```

```
SUB initial(A, v, lambda, dt)
  INPUT prompt "v = ": v              ! см/с
  LET A = 1                           ! амплитуда волны
  LET lambda = 2*pi                   ! (см)
  INPUT prompt "шаг по времени (с) между графиками = ": dt
END SUB

SUB screen(A, lambda, dt, xmax, ntime, space)
  LET xmax = 6*lambda
  LET ntime = 5  ! число моментов времени, для которых рисуем волну
  LET space = 0.5*A                   ! промежуток между графиками
  LET ymax = ntime*(space + 2*A) + 1
  SET WINDOW -xmax, 1.4*xmax, 0, ymax
  LET dx = lambda/4
  PRINT "расстояние между делениями = "; dx
  LET dy = A/10                       ! длина деления
  LET row = A
  LET t = 0
  LET ntick = xmax/dx
  FOR itime = 1 to ntime              ! ось x проводим ntime раз
    PLOT LINES: -xmax, row; xmax, row
    PLOT TEXT, at xmax, row: using$(" t =##.##", t)
    FOR itick = -ntick to ntick
      PLOT LINES: itick*dx, row; itick*dx, row + dy   ! рисуем деления
    NEXT itick
    LET row = row + space + 2*A
    LET t = t + dt
  NEXT itime
END SUB

SUB wavemotion(A, v, xmax, ntime, dt, space) ! графики смещения для 5 времен
  DECLARE DEF u
  LET npoint = 200
  LET dx = xmax/npoint
  LET row = A                         ! координата y на экране для u = 0
  FOR itime = 1 to ntime
    LET t = t + dt                    ! время
    FOR ipoint = -npoint to npoint
      LET x = ipoint*dx
      PLOT x, u(A, v, x, t) + row;
```

```
      NEXT ipoint
      PLOT
      LET row = row + space + 2*A
    NEXT itime
END SUB

DEF u(A, v, x, t)
  LET u = A*cos(x - v*t)
END DEF
```

ЗАДАЧА 8.8. Скорость волн

а. Программа **waves** рисует профиль $u = A\cos(x - vt)$ для множества значений x и при *ntime* = 5 различных временах. Задайте параметр $v = 1$ и определите скорость волны, измеряя расстояние, на которое смещается пик за промежуток времени t между картинками. В каком направлении движется волна при $v > 0$? Задайте $v = -1$ и определите направление движения волны.

б. Замените функцию $\cos(x - vt)$ на $\exp[-(x - vt)^2]$ и ответьте на те же вопросы, что и в п.«а».

Какими параметрами характеризуется гармоническая волна? Рассмотрим выражение $u(x,t) = A\cos(kx - \omega t)$, где ω и k — параметры. (Не следует путать параметр k с силовой постоянной.) Данная функция удовлетворяет волновому уравнению, если $\omega/k = v$. Чтобы понять смысл k и ω, положим сначала $t = 0$, так что $u(x) = A\cos kx$. Поскольку косинус есть периодическая функция своего аргумента, расстояние λ между двумя минимумами u определяется равенством $k\lambda = 2\pi$, т.е. $\lambda = 2\pi/k$. Это расстояние называется *длиной волны*, а величина $k = 2\pi/\lambda$ *волновым числом*. Аналогичные рассуждения для фиксированной координаты приводят к соотношению $T\omega = 2\pi$, где T — *период*. Поскольку частота $f = 1/T$, найдем, что $f = \omega/2\pi$, и отсюда $v = f\lambda$. В дальнейшем для описания гармонических волн мы будем часто пользоваться параметрами k и ω, а не f и λ.

ЗАДАЧА 8.9. Суперпозиция волн

а. С помощью программы **waves** понаблюдайте распространение волны, описываемой функцией

$$u(x,t) = \frac{4}{\pi}\,[1 + \sin(x-vt) + \frac{1}{3}\sin 3(x-vt) + \frac{1}{5}\sin 5(x-vt)]. \qquad (8.17)$$

(В программе **waves** необходимо только поменять описание внешней функции.) Какие значения волнового числа k представлены в формуле (8.17)? Опишите движение u.

б. С помощью программы **waves** понаблюдайте распространение волны, описываемой функцией

$$u(x,t) = \sin(x - vt) + \sin(x + vt) \qquad (8.18)$$

с $v = 1$ см/с и $2\pi/\lambda = 1$. Опишите получающуюся в результате волну.

в. Рассмотрите случай двух гармонических волн одинаковой амплитуды, которые обе распространяются в положительном направлении оси x, но имеют разные частоты ω_1 и ω_2. Примите, что для обеих волн $\omega = kv$, где скорость $v = 1$ см/с. Сумма волн принимает вид $u(x,t) = A[\sin(k_1x - \omega_1t) + \sin(k_2x - \omega_2t)]$. Примите $k_1 = 1.0$, $k_2 = 1.05$ и модифицируйте программу **waves**, чтобы проследить за $u(x,t)$ для разных времен. Опишите качественно форму $u(x,t)$ для фиксированного t. Каково расстояние между модуляциями амплитуды? Оцените *фазовую скорость*, т.е. скорость мелкомасштабных горбов амплитуды. Огибающая или пакет, образуемый несколькими группами волн, движется с *групповой скоростью*. Оцените групповую скорость для этого случая. Сравните величины фазовой и групповой скоростей.

В задаче 8.9 волны самых различных профилей сохраняли свою форму с течением времени. Такое свойство волны, которая, как говорят, *не диспергирует*, обусловлено линейностью связи ω и k, иначе говоря, каждая гармоника такой волны движется с одной и той же скоростью. Если же скорость распространения зависит от длины волны (или волнового числа), то говорят, что волна *диспергирует*, и в этом случае форма волны меняется со временем. Проиллюстрируем это явление в следующей задаче.

ЗАДАЧА 8.10. Дисперсия волн

а. Модифицируйте программу **waves**, чтобы рассмотреть распространение волнового профиля (8.17) через диспергирующую среду, в ко-

торой скорость зависит от k по закону $v(k) = v_0(1.1 + 0.1k)$ и $\omega(k) = v(k)k$. Модифицируйте выражение для $u(x,t)$, учтя дисперсию. Для простоты задайте $v_0 = 1$. Сохраняет ли волна свою форму? Вычислите фазовую скорость, находя расстояния, проходимые данным «горбом» амплитуды. Вычислите также групповую скорость, измеряя расстояние, проходимое огибающей волны. Сравните величины фазовой и групповой скоростей. Обратите внимание на то, что все решения волнового уравнения (8.12) имеют постоянные скорости распространения, не зависящие от k.

б. Повторите тот же расчет для волнового профиля, рассмотренного в задаче 8.9б.

8.5. ИНТЕРФЕРЕНЦИЯ И ДИФРАКЦИЯ

Интерференция является одной из самых фундаментальных характеристик всех волновых явлений. Термином *интерференция* традиционно пользуются в случаях, когда говорят о смешивании относительно небольшого числа источников волн, приходящих по отдельности от одного и того же источника. Однако термин *дифракция* имеет такой же смысл и обычно используется при большом числе источников. Поскольку наблюдать явления интерференции и дифракции относительно легко со светом, мы будем их рассматривать, ориентируясь на данную предметную область.

Классическим примером интерференции света является опыт Юнга с двойной щелью (рис. 8.4). Представим себе две узкие параллельные

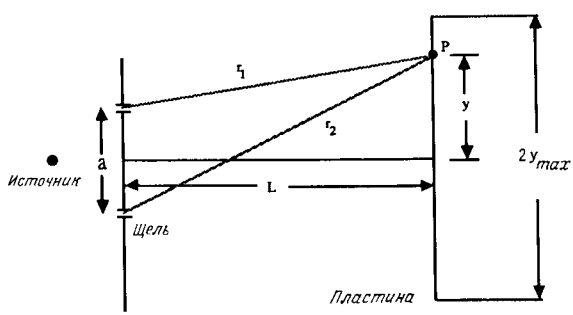

Рис. 8.4. Опыт Юнга с двойной щелью. Используемые в программе **interfere** параметры a, L и $ymax$ определены на рисунке.

щели, отстоящие друг от друга на расстоянии a и освещаемые источником, который излучает свет только одной частоты (монохроматический свет). Если такой световой источник помещен на одинаковом расстоянии от обеих щелей и отверстие щелей очень узкое, обе щели становятся когерентными источниками света с одинаковой фазой. Пока мы будем игнорировать тот факт, что щели действуют как линейные источники света, и будем считать их точечными источниками, например булавочными проколами. На расстоянии L располагается экран, на котором воспроизводятся интенсивности света от обоих источников. Что же мы видим на этом экране?

Излучаемый монохроматическим точечным источником свет представляет собой сферическую волну вида

$$E(r,t) = \frac{A}{r}\cos(kr - \omega t + \phi). \tag{8.19}$$

Множитель $1/r$ в (8.19) отражает тот факт, что интенсивность света убывает с расстоянием от источника. Из принципа суперпозиции известно, что полное электрическое поле в точке P (рис. 8.4) равно

$$E = E_1 + E_2 = \frac{A}{r_1}\cos(kr_1 - \omega t) + \frac{A}{r_2}\cos(kr_2 - \omega t). \tag{8.20}$$

Наблюдаемая интенсивность пропорциональна среднему по времени значению $|E|^2$.

Программа **interfere** выдает на экран дисплея интерференционную картину, обусловленную светом, испущенным N булавочными проколами. Предполагается, что каждая щель является источником гармонической волны одной и той же амплитуды, распространяющейся от нее во всех направлениях. Принятые в программе обозначения и ее основные особенности сводятся вкратце к следующему:

1. Программа выдает зависимость средней по времени интенсивности от расстояния y, представляющего собой координату вдоль экрана (см. рис. 8.4). Сложнее всего часть программы, где описаны параметры для вычерчивания графика. Входной параметр *ymax* задает максимальную координату экрана, для которой надо рисовать интенсивность.
2. В случае гармонической волны вида $E(x,t) = A\cos(\omega t+\phi)$ наблюдаемая интенсивность равна $(1/2)A^2$, где множитель $1/2$ возникает из-за усреднения по времени. Вместо усреднения по непрерывной

области значений t мы усредняем интенсивность по конечному набору значений $\omega t = 0,\ 2\pi/3$ и $4\pi/3$. Легко доказать, что при этом перекрестные члены взаимно уничтожаются и сумма $[\cos^2(0+\phi) + \cos^2(2\pi/3+\phi) + \cos^2(4\pi/3+\phi)]/3 = 1/2$, т.е. не зависит от ϕ. Данное усреднение производится в подпрограмме **pattern**.

```
PROGRAM interfere
CALL initial(nslit, a, L, lambda, ymax, nbar)
CALL pattern(nslit, a, L, lambda, ymax, nbar)
END

SUB initial(nslit, a, L, lambda, ymax, nbar)
  INPUT prompt "количество щелей = " : nslit
  INPUT prompt "расстояние между щелями (мм) = ": a
  INPUT prompt "расстояние до экрана (мм) = ": L
  INPUT prompt "длина волны (ангстремы)= ": lambda
  INPUT prompt "макс. координата на фотопластинке (мм) = ": ymax
  INPUT prompt "число точек усреднения интенсивности = ": nbar
  LET lambda = lambda*1e-7          ! перевод ангстремов в миллиметры
  LET Imax = (nslit/L)^2            ! максимальное значение интенсивности
  SET WINDOW -1.1*ymax, 1.1*ymax, -0.1*Imax, 1.1*Imax
  PLOT LINES: -ymax, 0; ymax, 0     ! проводим горизонтальную ось
  PLOT LINES: 0, Imax; 0, 0         ! проводим ось интенсивности
  PRINT "зависимость интенсивности от вертикальной координаты экрана"
  LET dy = lambda*L/a      ! расстояние между максимумами, если L >> a
  LET ntick = int(ymax/dy)
  LET Lt = 0.01*Imax                ! размер вертикального деления оси
  FOR itick = -ntick to ntick
    PLOT LINES: i*dy, 0; i*dy, Lt
  NEXT itick
END SUB

SUB pattern(nslit, a, L, lambda, ymax, nbar)
  LET k = 2*pi/lambda               ! волновое число
  LET npoint = 200                  ! число точек на полуоси y > 0
  LET dy = ymax/npoint              ! шаг рисования точек на экране
  LET delta = a/(nslit - 1)         ! расстояние между щелями
  LET phase = 2*pi/nbar
  LET L2 = L*L
```

```
    FOR ipoint = -npoint to npoint
      LET y = ipoint*dy              ! координата на экране
      LET intensity = 0              ! интенсивность света
      FOR t = 1 to nbar       ! усредняем интенсивность, а не амплитуду
        LET amplitude = 0
        FOR islit = 1 to nslit
          LET yslit = -0.5*a + (islit - 1)*delta      ! координата щели
          LET yslit = yslit - y  ! расстояние от щели до точки экрана по y
          LET r2 = L2 + yslit*yslit
          LET r = sqr(r2)          ! расстояние от щели до точки экрана
          LET amplitude = (1/r)*cos(k*r - phase*t) + amplitude
        NEXT islit
        LET intensity = intensity + amplitude*amplitude
      NEXT t
      LET intensity = intensity/nbar
      PLOT y,intensity;
    NEXT ipoint
  END SUB
```

ЗАДАЧА 8.11. Интерференция на двойной щели

а. С помощью программы **interfere** найдите распределение интенсивности света на экране, находящемся на расстоянии L от двойной щели. Пусть расстояние между щелями равняется a и y—координата вдоль экрана. Задайте L = 200 мм, a = 0.1 мм, $ymax$ = 5.0 мм и длину волны света λ = 500 Å. Программа переводит ангстремы в миллиметры (1 Å = 10^{-7}мм). Величина *nbar* равна числу моментов времени, по которым усредняется интенсивность. Получите интерференционную картину для *nbar* = 1 и 2. Почему для этих значений *nbar* картина сильно изрезана? Далее задайте *nbar* = 3 и посмотрите на эту картину. Сильно ли меняется картина, если *nbar* больше 3? Применяемая в программе **interfere** простая процедура усреднения дает тот же результат, что и настоящее усреднение по времени, происходящее в реальном эксперименте.

б. Пройдите по λ от 4000 до 6000 Å с постоянным шагом. Как меняется положение минимума в зависимости от λ? Что происходит, если λ = 1 Å?

в. При фиксированном λ проварьируйте L от 1 до 100 мм. Как зави-

сит положение максимума интенсивности от L, если $L \gg a$?

Первоначально одной из задач, стоящих при создании установки с двойной щелью, было измерение длины волны света. Однако гораздо лучше применять для той же цели метод *дифракционной решетки*, представляющей собой большое число равноотстоящих параллельных щелей. Простейшая решетка состоит из царапин, аккуратно нанесенных на плоскую стеклянную пластинку. Мы изучим картину интенсивности от такой решетки в задаче 8.12.

ЗАДАЧА 8.12. Дифракция на системе целей

Используйте программу **interfere** с параметрами $\lambda = 5000$ Å, $a = 0.01$ мм, $L = 200$ мм, *ymax* = 15 мм и *nbar* = 3. Рассмотрите случаи, когда N (число щелей) равно 3, 4, 5 и 10. Как меняется интенсивность пиков с N? Изменяется ли расстояние между пиками?

При рассмотрении опыта Юнга с двойной щелью и дифракционной решеткой мы предполагали, что каждая щель представляет собой булавочный прокол, который излучает только одну сферическую волну. На практике ширина реальных щелей гораздо больше длины волны видимого света. В задаче 8.13 мы рассмотрим картину света, получающуюся при падении плоской волны на отверстие, такое как одна щель. Мы используем *принцип Гюйгенса* и заменяем щель N когерентными источниками сферических волн. Такая замена является приближенной, но она применима, когда ширина отверстия велика по сравнению с длиной волны.

ЗАДАЧА 8.13. Дифракция на одной щели

а. Рассчитайте интенсивность света с длиной волны $\lambda = 5000$ Å от одной щели шириной 0.02 мм, заменяя ее $N = 20$ точечными источниками, отстоящими друг от друга на 0.001 мм. Задайте $L = 200$ мм, *nbar* = 3 и *ymax* = 30 мм и определите ширину центрального пика интенсивности. Как соотносится ширина центрального пика с шириной щели? Меняются ли ваши результаты с увеличением N?

б. Получите зависимость положения первого минимума дифракционной картины от длины волны, ширины щели и расстояния до экрана.

в. Получите дифракционную картину для $L = 1$ мм и 50 мм. Чем различаются эти картины?

ЗАДАЧА 8.14. Более реалистичное моделирование двойной щели

В данной задаче мы еще раз рассмотрим распределение интенсивности для интерференции на двойной щели, используя щели конечной ширины. Модифицируйте программу **interfer** и промоделируйте две «толстые» щели, заменяя каждую щель 20 точечными источниками, отстоящими друг от друга на 0.001 мм. Расстояние между серединами толстых щелей задайте равным 0.1 мм. Как соотносится данная картина интенсивности с картинами для одной и двойной щели? Как меняется эта картина в зависимости от длины волны?

*ЗАДАЧА 8.15. Дифракционная картина от прямоугольного отверстия

Аналогичный подход мы можем использовать для определения дифракционной картины от отверстия конечной ширины и высоты. Простейший метод состоит в том, что отверстие разбивается на маленькие квадраты и каждый квадрат рассматривается как источник сферической волны. Точно так же можно разбить экран или фотографическую пластинку на малые области, или ячейки, и вычислять среднюю по времени интенсивность в центре каждой ячейки. Необходимые вычисления просты, но занимают много времени из-за необходимости много раз вычислять косинус. Не столь понятна та часть задачи, которая связана с выбором способа представления различных значений вычисленной интенсивности на экране компьютера. Один из возможных методов состоит в изображении в случайных местах каждой ячейки «точек», причем их количество берется пропорциональным рассчитанной в центре ячейки интенсивности. Рекомендуемые параметры: $\lambda = 5000$ Å, $L = 200$ мм для отверстия размером 1×3 мм.

8.6. ПОЛЯРИЗАЦИЯ

До сих пор нас не интересовало направление колебаний волны. Например, использованное в разд.8.1 описание продольных колебаний линейной цепочки с равным успехом годилось и для поперечных колебаний струны. Теперь мы рассмотрим некоторые явления, зависящие от *поляризации* волны.

Задание направления распространения продольной волны автоматически определяет направление колебаний среды. Что касается попереч-

ной волны, то единственным ограничением направления колебаний является то, что они должны происходить в плоскости, перпендикулярной направлению распространения. Рассмотрим поперечные колебания электрического поля в электромагнитной волне, распространяющейся в направлении оси z. Электрическое поле может быть представлено двумерной векторной функцией $\mathbf{E}(z,t)$ с компонентами $E_x(z,t)$ и $E_y(z,t)$. В случае идеального монохроматического света электрическое поле колеблется с одной определенной частотой. Однако x- и y-компоненты вектора E колеблются независимо и могут быть представлены в виде

$$E_x(k,t) = E_{x0}\cos(kz - \omega t - \phi), \tag{8.21}$$

$$E_y(k,t) = E_{y0}\cos(kz - \omega t), \tag{8.22}$$

где E_{x0} и E_{y0} — соответствующие амплитуды, а ϕ — относительная фаза обеих компонент. Чтобы найти электрическое поле, образуемое компонентами (8.21) и (8.22), сложим векторно E_x и E_y и найдем результирующее поле **E**.

Далее воспользуемся программой **polarize**, которая поможет нам наглядно представить основные свойства волны **E**. Основная часть этой программы демонстрирует временну́ю зависимость величин E_x, E_y и **E** для «входящей» в точке $z = 0$ волны. На четвертом графике изображается «выходящая» волна $\mathbf{E}(z,t)$ при $z > 0$. Этот последний график будет использоваться в задаче 8.17 для изображения результата прохождения света через двоякопреломляющее вещество.

Поскольку программа **polarize** является демонстрационной, организация вывода результатов в большей степени зависит в ней от особенностей аппаратуры и программного обеспечения. Например, для изображения величин E_x и E_y мы использовали имеющиеся в языке True BASIC средства мультипликации. Поскольку в нашей версии программы «наконечник» вектора рисуется в виде прямоугольника, а не стрелки или треугольника, вы, возможно, пожелаете переписать эту программу. С другой стороны, не забудьте, что наша цель — изучение физики, а не построение идеального изображения на экране. Основные особенности программы включают в себя следующее:

1. Горизонтальная и вертикальная линии проводятся с целью обозначения диапазона возможных значений E_x и E_y. Значения E_x и E_y вычисляются по формулам (8.21) и (8.22), а их «конец» вычерчивается у текущего значения E_x и E_y.

2. Векторная сумма E_x и E_y изображается в виде вращающегося отрезка, образующего с горизонтальной осью угол θ, определяемый выражением $\mathrm{tg}\,\theta = E_y/E_x$. Предыдущее положение вращающегося отрезка не стирается перед рисованием нового отрезка, относящегося к моменту времени на Δt более позднему. Возможно, вы пожелаете модифицировать программу так, чтобы бóльшая часть отрезка стиралась и оставлялся лишь конец отрезка, чтобы передать движение конца вектора электрического поля.

Предполагается, что, прежде чем пытаться что бы то ни было менять в программе, вы сначала выполните ее и посмотрите, как выглядит изображение на экране.

```
PROGRAM polarize
CALL initial(Ex, Eγ, phase, width, nx, nγ, lambda, γmax, tip$, erase$)
CALL polar(Ex, Eγ, phase, width, nx, nγ, lambda, γmax, tip$, erase$)
END

SUB initial(Ex0, Eγ0, phase, width, nx, nγ, lambda, γmax, tip$, erase$)
  INPUT prompt "отношение (>=1) Ex к Eγ = ": ratio
  INPUT prompt "разность фаз (радианы) = ": phase
  ! следующие три инструкции нужны для задачи 8.17
  ! INPUT prompt "ширина кристалла (мм) = ": width
  ! INPUT prompt "показатель преломления кристалла по x = ": nx
  ! INPUT prompt "показатель преломления кристалла по γ = ": nγ
  LET lambda = 5.4e-4              ! длина волны (мм)
  LET Ex0 = ratio
  LET Eγ0 = 1
  LET aspect_ratio = 1.5           ! зависит от типа монитора
  LET Lmax = 10*ratio              ! размер по горизонтали
  LET γmax = Lmax/aspect_ratio
  SET WINDOW -1, Lmax, -0.5*γmax, 0.5*γmax
  PLOT TEXT, at 0.1*Lmax, 0.45*γmax: "входящая волна"
  ! PLOT TEXT, at 0.8*Lmax, 0.45*γmax: "выходящая волна" ! Задача 8.17
  LET r = 0.1  ! линейный размер "конца" вектора, изображающего Ex и Eγ
  BOX AREA 1,1 + r,1,1 + r
  BOX KEEP 1,1 + r,1,1 + r in tip$
  BOX CLEAR 1,1 + r,1,1 + r
  BOX KEEP 1,1 + r,1,1 + r in erase$
END SUB
```

```
SUB polar(Ex0, Eγ0, phase, width, nx, nγ, lambda, γmax, tip$, erase$)
  LET spacing = 2.5*Ex0    ! расстояние между рисунками по горизонтали
  LET Lx = 0.9*Ex0         ! горизонтальная координата вычерчивания Ex
  PLOT TEXT, at Lx, 0.25*γmax: "Ex"
  LET Lγ = Lx + spacing
  PLOT TEXT, at Lγ, 0.25*γmax: "Eγ"
  LET Lin = Lγ + spacing
  PLOT TEXT, at Lin, 0.25*γmax: "E вх."
  LET Lout = Lin + spacing
  ! PLOT TEXT, at Lout, 0.25*γmax: "E вых."                    ! Задача 8.17
  LET t = 0                          ! время
  LET k = 2*pi/lambda                ! волновой вектор
  LET c = 3e11                       ! скорость света (мм/с)
  LET w = k*c                        ! угловая частота
  LET dt = 0.1/w
  DO
    BOX SHOW erase$ at Lx + Ex, 0
    PLOT LINES: Lx-Ex0, 0; Lx+Ex0, 0  ! проводим линию от -Ex0 до +Ex0
    BOX SHOW erase$ at Lγ, Eγ
    PLOT LINES: Lγ, -Eγ0; Lγ, Eγ0     ! проводим линию от -Eγ0 до +Eγ0
    LET t = t + dt
    LET Ex = Ex0*cos(w*t - phase)
    LET Eγ = Eγ0*cos(w*t)
    ! LET Exout = Ex0*cos(w*t - k*width/nx - phase)              ! Задача 8.17
    ! LET Eγout = Eγ0*cos(w*t - k*width/nγ)                      ! Задача 8.17
    BOX SHOW tip$ at Lx+Ex, 0
    BOX SHOW tip$ at Lγ, Eγ
    PLOT LINES: Lin, 0; Lin + Ex, Eγ
    ! PLOT LINES: Lout, 0; Lout + Exout, Eγout                   ! Задача 8.17
  LOOP UNTIL key$ input
END SUB
```

В задаче 8.16 мы используем программу **polarize** для исследования различных типов поляризованных волн.

ЗАДАЧА 8.16. Поляризация

а. В программе **polarize** величина E_{y0} задается равной единице, а значение E_{x0} определяется отношением $ratio = E_{x0}/E_{y0}$. Возьмите

$E_{x0} = E_{y0}$ и $\phi = 0$ и определите тип результирующего вектора электрического поля. Почему говорят, что в данном случае результирующее колебание *линейно поляризовано*? Как будет поляризована волна, если $E_{x0} = 2$, $E_{y0} = 1$ и $\phi = 0$?

б. Если конец вектора электрического поля описывает окружность, то говорят, что свет имеет *круговую поляризацию*. Получите такую поляризацию, взяв $E_{x0} = E_{y0} = 1$ и $\phi = \pi/2$. Если конец вектора электрического поля движется против часовой стрелки (если смотреть со стороны положительного направления оси z), свет имеет правую круговую поляризацию. Каково направление поляризации для $\phi = \pi/2$? При каком значении ϕ получается волна с левой круговой поляризацией?

в. Какого типа волна получается для $E_{x0} = 2$, $E_{y0} = 1$ и $\phi = \pi/2$? Какой тип получается при произвольных относительных значениях E_{x0}, E_{y0} и ϕ?

г. В программе **polarize** длина волны задана равной 5400 Å. Поварьируйте длину волны и определите, влияет ли длина волны на тип получающегося вектора **E**.

В некоторых веществах — таких как кварц и кальцит — скорость волны зависит от направления колебания вектора электрического поля. Такие вещества называются *веществами с двойным лучепреломлением*, или *двоякопреломляющими*. Как показано в задаче 8.17, с помощью этих веществ можно изменить состояние поляризации проходящего через них луча (рис. 8.5).

ЗАДАЧА 8.17. Пластинка в четверть волны

Модифицируйте программу **polarize**, удалив восклицательные знаки (!) в первых инструкциях, которые определяют изменение состояния поляризации светового луча, проходящего через вещество с двойным лучепреломлением. Задайте показатель преломления в направлении оси x равным 1.2, а показатель преломления света в направлении оси y равным 1.1. Предположите, что падающий на кристалл свет поляризован по кругу. Найдите, при какой толщине кристалла *width* (в миллиметрах) свет, имеющий круговую поляризацию, на выходе из него становится линейно поляризованным. Что получается, если че-

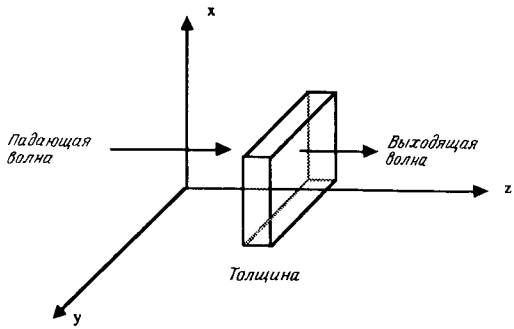

Рис. 8.5. Вещество с двойным лучепреломлением имеет разные показатели преломления по осям x и y.

рез кристалл той же толщины проходит линейно поляризованный свет? Вещество с двойным лучепреломлением такой толщины называется пластинкой «в четверть волны» (или пластинкой $\lambda/4$).

8.7. ГЕОМЕТРИЧЕСКАЯ ОПТИКА И ПРИНЦИП НАИМЕНЬШЕГО ВРЕМЕНИ

Наш повседневный опыт со светом приводит естественным образом к понятию световых лучей, распространяющихся прямолинейно и отражающихся или преломляющихся по геометрическим законам. Это описание распространения света, называемое *геометрической*, или *лучевой*, оптикой, применимо, когда длина волны света мала по сравнению с линейными размерами любых препятствий или детекторов, которые мы могли бы использовать. Геометрическая оптика рассматривает явления отражения и преломления, а также построение систем линз, таких как телескопы и микроскопы.

Распространение световых лучей подчиняется простому принципу, установленному Ферма: *луч света идет по такому пути между двумя точками (независимо от любых прочих условий), который требует наименьшего времени.* Принцип Ферма, или принцип наименьшего времени, можно принять в качестве основы геометрической оптики. Данный прин-

цип — не единственный экстремальный принцип в физике. Аналогичный принцип, известный как *принцип наименьшего действия*, может быть использован вместо законов движения Ньютона в качестве фундамента всей классической механики.

Одно простое применение принципа наименьшего времени касается пути, проходимого светом из точки A в точку B в одной и той же среде. Поскольку скорость в любом направлении одинакова, путь за наименьшее время — это путь по кратчайшему расстоянию, т.е. по прямой, соединяющей точку A с B. В результате мы приходим к довольно очевидному заключению, что в однородной среде свет распространяется прямолинейно. Покажите, что будет, если наложить условие, чтобы свет, прежде чем достигнуть B, обязательно попадал на зеркало.

Интереснее применения принципа наименьшего времени к задачам преломления, где свет падает на поверхность раздела двух веществ, в которых скорость света различна. Принято выражать скорость света в среде v через скорость света в вакууме c и *показатель преломления* n среды

$$n = \frac{c}{v}. \tag{8.23}$$

Показатели преломления некоторых распространенных веществ приведены в табл. 8.2.

ТАБЛИЦА 8.2. Показатели преломления видимого света для некоторых распространенных веществ

Вещество	Показатель преломления
Воздух	1.0003
Вода	1.33
Стекло	1.5
Бриллиант	2.4

В задачах 8.18 и 8.19 мы рассматриваем некоторые следствия из принципа наименьшего времени. Наша задача состоит в том, чтобы разработать программу для ЭВМ, которая позволит рисовать различные пути и находить методом проб и ошибок путь с наименьшим временем. Некоторые особенности программы **Fermat** заключаются в следующем:

1. Считается, что свет распространяется слева направо через N сред (рис. 8.6).

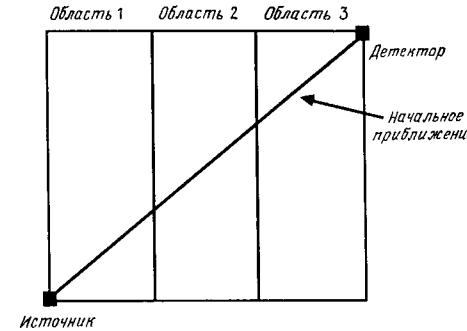

Рис. 8.6. Вид изображения на экране для программы **Fermat**.

2. Координаты «источника» света и «детектора» равны $\{0, y(0)\}$ и $\{N, y(N)\}$ соответственно, где $y(0) = -y(N)$.
3. Толщина каждой области равна единице и показатель преломления в каждой области однороден. Имеется $N-1$ граница, разделяющая N сред, при этом показатель преломления i-й среды $n(i)$ определяется по формуле $n(i) = 1 + (i-1)*dn$. (Индекс i растет слева направо.) Скорость света в вакууме положена равной единице.
4. Поскольку в каждой среде свет распространяется прямолинейно, путь света определяется координатой $y(i)$ на каждой границе.
5. В качестве начального приближения для пути берется прямая линия, соединяющая обе заданные концевые точки. Изменение пути света осуществляется посредством изменения значения $y(i)$ в точке пересечения пути с определенной границей двух сред. Для изменения значения $y(i)$ курсор перемещают влево или вправо, пока не будет достигнута требуемая граница, и затем двигают курсор вверх или вниз.
6. Все четыре возможных перемещения курсора рассматриваются с помощью структуры выбора **SELECT CASE**, которая эквивалентна последовательности инструкций else-if. Переменные, записаные в числовой форме, суть ASCII-коды, отвечающие той или иной нажимаемой на клавиатуре клавише. Если нажата клавиша 'u' или 'd', точка на пути перемещается вверх (up) или вниз (down) на величину *delta*. Аналогично нажатие клавиш 'l' или 'r' перемещает курсор соответственно влево (left) или вправо (right). Ввод с клавиатуры 's' (stop) останавливает программу.

7. В тот момент, когда путь изменен, рисуется новый путь, а также текущий «наилучший» путь. Время текущего минимального пути и время пробного пути выводятся на экран.

```
PROGRAM Fermat
DIM y(0 to 30),ybest(0 to 30)
CALL initial(y,ybest,N,tpath,tbest,dn,delta,cursor$,rx,ry)
CALL path(y,ybest,N,tpath,tbest,dn,delta,cursor$,rx,ry)
END

SUB initial(y(),ybest(),N,tpath,tbest,dn,delta,cursor$,rx,ry)
  INPUT prompt "количество областей = ":N
  INPUT prompt "приращение показателя преломления = ": dn
  INPUT prompt "y-координата детектора = ": y(N)
  LET y(0) = -y(N)                    ! координата источника
  LET ybest(0) = y(0)                 ! координаты неподвижных концов
  LET ybest(N) = y(N)
  LET slope = (y(N) - y(0))/(N) ! угловой коэффициент начального пути
  FOR i = 1 to N
    LET y(i) = y(0) + slope*i
    ! начальное приближение - прямая, соединяющая источник с детектором
    LET ybest(i) = y(i)
  NEXT i
  ! толщина каждой области равна 1
  SET window -1,N + 1,y(0),y(N)
  BOX LINES 0,N,y(0),y(N)             ! рисуем рамку вокруг области
  FOR i = 1 to (N - 1)
    ! отделяем области вертикальными линиями
    PLOT LINES: i,y(0); i,y(N)
  NEXT i
  LET ry = 0.1
  LET rx = 0.02
  ! детектор и источник изображаем квадратиками
  BOX AREA - rx,rx,y(0) - ry,y(0) + ry
  BOX AREA N - rx,N + rx,y(N) - ry, y(N) + ry
  LET ry = 0.025
  LET rx = 0.01
  BOX KEEP N - rx,N + rx, y(N) - ry,y(N) + ry in cursor$
  FOR i = 1 to N
```

```
    PLOT LINES: i,y(i); i-1,y(i-1)
    PLOT LINES: i,ybest(i); i-1,ybest(i-1)
    LET dy2 = (y(i) - y(i-1))*(y(i) - y(i-1))
    LET length = sqr(1 + dy2)        ! длина пути
    LET tpath = tpath + length*(1 + (i-1)*dn)
  NEXT i
  LET tbest = tpath
  ! изменение пути, приходящееся на одно нажатие клавиши
  LET delta = 0.01*abs(y(N) - y(0))
  PRINT using "время пробного пути =**.***": tpath
  PRINT using "наилучшее из всех вариантов время =**.***": tbest
END SUB

SUB path(y(), ybest(), N, tpath, tbest, dn, delta, cursor$, rx, ry)
  LET i = 1                          ! начинаем на границе областей 1 и 2
  BOX SHOW cursor$ at i - rx,y(i) - ry
  DO
    LET choice = 0
    IF key INPUT then                ! нажата клавиша u, d, l, r или s
      GET KEY choice
      SELECT CASE choice
      CASE 108                       ! переместить курсор влево
        IF i > 1 then
          LET deltax = -1
          CALL boundary(y, i, deltax, cursor$, rx, ry)
        END IF
      CASE 114                       ! переместить курсор вправо
        IF i < N - 1 then
          LET deltax = 1
          CALL boundary(y, i, deltax, cursor$, rx, ry)
        END IF
      CASE 117                       ! переместить курсор вверх
        CALL newpath(y, ybest, i, N, tpath, tbest, dn, delta, cursor$, rx, ry)
      CASE 100                       ! переместить курсор вниз
        CALL newpath(y, ybest, i, N, tpath, tbest, dn, -delta, cursor$, rx, ry)
      CASE ELSE ! если нажата клавиша, отличная от u, d, l, r или s
      END SELECT
    END IF
  LOOP until choice = ord("s")
END SUB
```

```
SUB boundary(y(), i, deltax, cursor$, rx, ry)
  BOX CLEAR i - rx,i + rx,  y(i) - ry,y(i) + ry          ! стираем курсор
  ! рисуем заново ранее частично стертую линию
  PLOT LINES: i,y(i) - ry; i,y(i) + ry
  LET i = i + deltax
  BOX SHOW cursor$ at i - rx,y(i) - ry
END SUB

SUB newpath(y(), ybest(), i, N, tpath, tbest, dn, delta, cursor$, rx, ry)
  CALL erase(i, y, rx, ry)
  LET y(i) = y(i) + delta
  CALL draw(i, y, ybest)
  BOX SHOW cursor$ at i - rx,y(i) - ry
  CALL pathtime(N, i, dn, delta, y, ybest, tpath, tbest)
END SUB

SUB erase(i, y(), rx, ry)
  BOX CLEAR i - rx,i + rx,y(i) - ry,y(i) + ry
  SET color "background"
  PLOT LINES: i,y(i); i - 1,y(i-1)
  PLOT LINES: i,y(i); i + 1,y(i+1)
END SUB

SUB draw(i, y(), ybest())
  SET color "white"
  PLOT LINES: i,y(i); i - 1,y(i-1)
  PLOT LINES: i,y(i); i + 1,y(i+1)
  PLOT LINES: i,ybest(i); i - 1,ybest(i-1)
  PLOT LINES: i,ybest(i); i + 1,ybest(i+1)
END SUB

SUB pathtime(N, i, dn, delta, y(), ybest(), tpath, tbest)
  LET v1 = 1/(1 + (i-1)*dn)          ! скорость света в области i - 1
  LET v2 = 1/(1 + i*dn)              ! скорость света в области i
  LET dy2 = (y(i) - y(i-1))*(y(i) - y(i - 1))
  LET t1 = sqr(1 + dy2)/v1           ! время пути в области i - 1
  LET dy2 = (y(i) - y(i+1))*(y(i) - y(i+1))
  LET t2 = sqr(1 + dy2)/v2           ! время пути в области i
  LET dy2 = (y(i) - delta - y(i-1))*(y(i) - delta - y(i-1))
  LET t1old = sqr(1 + dy2)/v1
  LET dy2 = (y(i) - delta - y(i+1))*(y(i) - delta - y(i+1))
```

```
    LET t2old = sqr(1 + dy2)/v2
    LET tpath = tpath + t1 + t2 - t1old - t2old
    IF tpath <= tbest then
      LET tbest = tpath
      FOR j = 1 to N
        SET color "background"
        ! стираем старый "наилучший" путь
        PLOT LINES: j,ybest(j); j - 1,ybest(j-1)
        SET color "white"
        PLOT LINES: j,y(j); j - 1,y(j-1)  ! рисуем новый наилучший путь
      NEXT j
      FOR j = 1 TO N
        LET ybest(j) = y(j)
      NEXT j
    END IF
    IF delta > 0 then
      PLOT LINES: i,y(i) - 2*delta; i,y(i) + delta
    ELSE
      PLOT LINES: i,y(i) + delta; i,y(i) - 2*delta
    END IF
    SET cursor 1,1
    PRINT using "время пробного пути =**.***": tpath
    PRINT using "наилучшее из всех вариантов время =**.***": tbest
  END SUB
```

ЗАДАЧА 8.18. Закон преломления

а. Используйте программу **Fermat** для нахождения угла падения θ_1 и угла преломления θ_2 на границе двух сред с различными показателями преломления. Углы θ_1 и θ_2 измеряются от нормали к границе. Задайте $N = 2$ и в качестве первой среды возьмите воду ($n \approx 1$), а в качестве второй — стекло ($n \approx 1.5$). Величину $y(N)$ — координату детектора — неплохо взять равной $y(N) = 3$. («Стандартное» значение координаты источника $y(0)$ равно $-y(N)$, а стандартное значение *delta* — приращения пути по вертикали — составляет 1% от $y(N) - y(0)$.) После нахождения пути с наименьшим временем, «скопируйте» экран и измерьте угол падения θ_1 и угол преломления θ_2.

б. Модифицируйте программу таким образом, чтобы первая среда представляла собой стекло, а вторая среда — воду ($n \approx 1.33$). Убедитесь, что ваши результаты, полученные в п.п. «а» и «б», согласуются с законом Снеллиуса: $n_2 \sin\theta_2 = n_1 \sin\theta_1$.

в. Каково относительное изменение времени пути для путей, проходящих вблизи «начального приближения»? Каково относительное изменение времени пути для путей, проходящих вблизи оптимального пути? В каком случае относительное изменение меньше? Как говорится у Фейнмана и др. (см. список литературы), более точная формулировка принципа Ферма следующая: свет выбирает один путь из множества близлежащих, требующих почти *одинакового* времени для прохождения.

ЗАДАЧА 8.19. Пространственно неоднородный показатель преломления

а. Чтобы приобрести опыт в отыскании пути минимального времени в случае большого числа границ, пропустите программу **Fermat** с $N = 3$ и 4, $y(N) = 3$ и $dn = 0.2$. Насколько быстро вы можете заставить свои пробные пути сойтись к наименьшему времени?

б. Предположим, имеется среда, у которой показатель преломления зависит от координаты. Примером может служить земная атмосфера, имеющая низкую плотность в верхних слоях и высокую у земной поверхности. Такую неоднородную среду можно моделировать, разбивая систему на слои одинаковой толщины, каждый из которых считается однородным. Показатель преломления i-й области равен $n(i) = 1 + (i - 1)*dn$. Пропустите программу **Fermat** для $N = 10$ и $dn = 0.01$ и найдите путь наименьшего времени. Опираясь на полученные результаты, объясните тот факт, что, когда мы наблюдаем заход Солнца, оно уже находится за горизонтом (рис. 8.7). Пример сходного явления — мираж, который можно наблюдать при езде по раскаленной дороге. Бывает, что поверхность дороги кажется «мокрой», хотя на самом деле она сухая. Поскольку воздух у поверхности дороги сильно нагрет, он имеет меньшую плотность, чем воздух, расположенный выше. Используйте свои результаты для объяснения природы такого миража.

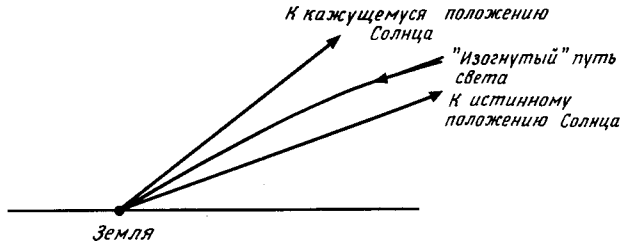

Рис. 8.7. У горизонта Солнце кажется (преувеличенно) выше, чем на самом деле.

ЛИТЕРАТУРА

Frank S. Crawford, Waves, Berkeley Physics Course, Vol. 3, McGraw-Hill, 1968. [Имеется перевод: *Ф. Крауфорд*, Волны. — М.: Мир, 1975] Блестящая книга о всех видах волн. Весьма рекомендуются домашние опыты. Одно наблюдение волновых явлений стоит многих демонстраций на компьютере.

N. A. Dodd, Computer simulation of diffraction patterns, Phys. Educ. **18**, 294 (1983).

Robert M. Eisberg, *Lawrence S. Lerner*, Physics, Vol.II, McGraw-Hill, 1981. Авторы этого вводного курса рассматривают принцип Ферма, а также традиционные вопросы по оптике и волнам.

David Falk, *Dieter Brill*, *David Stork*, Seeing the Light, Harper and Row, 1986. Прекрасно иллюстрированная книга по оптическим явлениям, написанная для не естественно-научных специальностей. Несмотря на упрощенный уровень книги, авторы рассмотрели принцип Бабине, который отсутствует во многих книгах по оптике промежуточного уровня.

Richard P. Feynman, *Robert B. Leighton*, *Matthew Sands*, The Feynman Lectures on Physics, Vol.1, Addison-Wesley (1963). [Имеется перевод: *Р. Фейнман, Р. Лейтон* и др., Фейнмановские лекции по физике. — М.: Мир, 1966.] В гл. 26 отлично рассмотрен принцип наимень-

шего времени. Кроме того, вопросы, связанные с волновыми явлениями, рассматриваются в гл. 28—30 и 33.

A. P. French, Vibrations and Waves, W.W.Norton & Co., 1971. Учебник вводного уровня, в котором основное внимание уделено механическим системам.

Akira Hirose, Karl E. Lonngren, Introduction to Wave Phenomena, John Wiley & Sons, 1985. Учебник промежуточного уровня, в котором рассматриваются общие свойства волн в различных средах.

F. Graham Smith, J. H. Thomson, Optics, John Wiley & Sons, 1971. Учебник промежуточного уровня по оптике, в котором изложение ведется на основе волнового формализма.

Garrison Sposito, An Introduction to Classical Dynamics, John Wiley & Sons, 1976. В гл. 6 рассмотрена задача о связанных гармонических осцилляторах.

Michael L.Williams, Humphrey J.Maris, Numerical study of phonon localization in disordered systems, Phys. Rev. **B31**, 4508 (1985). Авторы открывают заново алгоритм, которому мы присвоили имя Эйлера—Кромера, и применяют его для получения нормальных колебаний двумерной системы связанных осцилляторов со случайными массами.

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

Пирс Дж., Почти все о волнах.—М.: Мир, 1976. Книга, в которой рассматриваются вопросы на том же уровне, что и в этой главе.

Сивухин Д. В., Оптика.—М.: Наука, 1985. Подробно рассмотрены принцип Ферма, интерференция и дифракция волн, поляризация света.

Косевич А. М., Ковалев А. С., Введение в нелинейную физическую механику.—Киев: Наукова думка, 1989. В гл. 2—3 подробно исследуется движение системы связанных осцилляторов.

СТАТИЧЕСКИЕ ПОЛЯ ЗАРЯДОВ И ТОКОВ

9

Вычисляются электрическое и магнитное поля, создаваемые стационарными распределениями зарядов и токов. Кроме того, рассматривается метод релаксации для получения численного решения уравнений Лапласа и Пуассона.

9.1. ВВЕДЕНИЕ

Известно, что на заряд q, движущийся со скоростью $\mathbf{v}$, действует сила Лоренца

$$\mathbf{F} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B}), \tag{9.1}$$

где $\mathbf{E}$ есть *электрическое поле*, а $\mathbf{B}$ — *магнитное поле* в точке нахождения заряда. Следовательно, если известны $\mathbf{E}$ и $\mathbf{B}$, то мы можем рассчитать движение заряженной частицы. В этой главе для вычисления полей $\mathbf{E}$ и $\mathbf{B}$, создаваемых стационарными распределениями зарядов и токов, мы используем несколько методов. Кроме того, для нахождения численных решений уравнений Лапласа и Пуассона мы применим метод релаксации.

9.2. ЭЛЕКТРИЧЕСКИЕ ПОЛЯ И ПОТЕНЦИАЛ

Предположим, мы хотим найти электрическое поле $\mathbf{E}(\mathbf{r})$ в точке $\mathbf{r}$, обусловленное точечными зарядами $q_1, q_2, \ldots, q_N$. Мы знаем, что $\mathbf{E}(\mathbf{r})$ удовлетворяет принципу суперпозиции и имеет вид

$$\mathbf{E}(\mathbf{r}) = K \sum_{i}^{N} \frac{q_i}{|\mathbf{r} - \mathbf{r}_i|^3} (\mathbf{r} - \mathbf{r}_i), \tag{9.2}$$

где $\mathbf{r}_i$ — координата неподвижного i-го заряда и K — постоянная, зависящая от выбора системы единиц. В системе СИ постоянная K равна

$$K = 1/4\pi\varepsilon_0 \approx 9.0 \cdot 10^9 \text{ Н} \cdot \text{м}^2/\text{Кл}^2, \tag{9.3}$$

где кулон (Кл) есть единица заряда в системе СИ, а ε_0 — диэлектрическая постоянная вакуума. С микроскопической точки зрения кулон — это очень большая единица заряда. Например, величина заряда электрона равняется $e \approx 1.6 \cdot 10^{-19}$ Кл. Поэтому можно предположить, что во многих модельных расчетах нам придется выбирать разные единицы.

Обратите внимание, что поле $\mathbf{E}$ является *векторным*. В каждой точке пространства такое поле характеризуется величиной и направлением. Как наглядно изобразить такое векторное поле? Один из возможных способов — разбить пространство дискретной сеткой, найти $\mathbf{E}$ в каждой

точке и начертить из этих точек стрелки в направлении **E**. Однако этот способ не дает никакой информации о величине электрического поля. Более хорошим способом наглядного представления векторного поля является изображение *силовых линий электрического поля*. Эти линии обладают следующими свойствами:

1. Каждая силовая линия электрического поля представляет собой направленную линию, касательная к которой в каждой точке параллельна электрическому полю в этой точке.
2. Эти линии гладкие и непрерывные за, исключением особых точек, таких как точечные заряды. (Бессмысленно говорить об электрическом поле *в точке* точечного заряда.)
3. Полное число электрических силовых линий, исходящих из точечного заряда, пропорционально величине этого заряда. Коэффициент пропорциональности выбирается из соображений наибольшей ясности изображения поля. Изображение силовых линий — это искусство плюс наука.

Нижеследующая программа **fieldline** вычерчивает силовые линии электрического поля в двумерном случае, используя следующий алгоритм:

1. Выбираем точку (x,y) и вычисляем компоненты E_x и E_y вектора электрического поля **E** по формуле (9.2).
2. Проводим в этой точке небольшой прямолинейный отрезок заданной длины Δs в направлении **E**. Компоненты этого отрезка равны

$$\Delta x = \Delta s \frac{E_x}{|\mathbf{E}|} \qquad \text{и} \qquad \Delta y = \Delta s \frac{E_y}{|\mathbf{E}|}.$$

3. Повторяем данную процедуру с новой точки $(x + \Delta x, y + \Delta y)$. Продолжаем до тех пор, пока силовая линия не уйдет в бесконечность или не подойдет к какому-нибудь отрицательному заряду.

Программа **fieldline** не обеспечивает автоматически правильную плотность линий. Вы должны сами выбрать каждую точку пространства, в которой компьютер начинает рисовать силовую линию. Чтобы получить правильную плотность линий, необходимо начинать около точек, где распределение силовых линий очевидно. Известно, например, что силовые линии всегда выходят из положительного заряда в радиальных на-

правлениях. Следовательно, следует начинать строить силовые линии вблизи положительного заряда таким образом, чтобы число силовых линий, начинающихся у каждого положительного заряда, было пропорционально величине заряда этой частицы. Например, если имеются заряды 2 мкКл и 4 мкКл, то число линий, начинающихся у заряда 4 мкКл, должно быть в два раза больше, чем берущих начало из заряда 2 мкКл. Курсор можно перемещать в требуемую точку экрана нажатием клавиш 'u', 'd', 'l' и 'r', означающих вверх (up), вниз (down), влево (left) и вправо (right). Для вычерчивания силовой линии, начинающейся в текущем положении курсора, нажмите клавишу 'p' (plot). Для прекращения рисования линии нажмите любую клавишу.

Основные особенности программы **fieldline** касаются ее графических инструкций. Символ курсора запоминается в строковой переменной при помощи инструкций **BOX AREA** и **BOX KEEP**. Переменная **save$** служит для запоминания части экрана, в которую предстоит выводить курсор. Поскольку в месте размещения курсора могут оказаться уже проведенные линии и курсор их сотрет, с помощью переменной **save$** эти линии перерисовываются после перемещения курсора в новое положение.

```
PROGRAM fieldline
DIM x(20), y(20), q(20)
CALL charges(N, x, y, q, ymax)                    ! ввод координат зарядов
CALL screen(N, x, y, q, xmax, ymax, dx, dy, r, cursor$, save$)
CALL move(N, x, y, q, xmax, ymax, dx, dy, r, cursor$, save$, xcursor, ycursor)
END

SUB charges(N, x(), y(), q(), ymax)          ! ввод координат и величин зарядов
   INPUT prompt"количество зарядов = ": N
   FOR i = 1 to N
      PRINT "заряд (микрокулоны) частицы ";i;
      INPUT q(i)
      PRINT "x и y координаты (метры) частицы ";i;
      INPUT x(i), y(i)
      IF xmax < abs(x(i)) then LET xmax = abs(x(i))
      IF ymax < abs(y(i)) then LET ymax = abs(y(i))
   NEXT i
   LET ymax = max(xmax, ymax)
   LET ymax = 1.5*ymax
   IF ymax = 0 then LET ymax = 2
END SUB
```

```
SUB screen(N, x( ), y( ), q( ), xmax, ymax, dx, dy, r, cursor$, save$)
    LET aspect_ratio = 1.5              ! зависит от типа монитора
    LET xmax = aspect_ratio*ymax
    SET window -xmax, xmax, -ymax, ymax
    LET r = 0.01*xmax
    FOR i = 1 to N
       ! заряды рисуем в виде окружностей радиусом r
       BOX CIRCLE x(i)-r, x(i)+r, y(i)-r, y(i)+r
       ! отрицательные заряды изображаем сплошными кружками
       IF q(i) < 0 then FLOOD x(i), y(i)
    NEXT i
    LET dx = 0.025*xmax  ! dx и dy-минимальные размеры шагов курсора
    LET dy = 0.025*ymax
    BOX KEEP -r, r, -r, r in save$
    LET xtemp = 0.9*xmax
    LET ytemp = 0.9*ymax
    ! задаем форму курсора
    BOX AREA xtemp-r, xtemp+r, ytemp-r, ytemp+r
    BOX KEEP xtemp-r, xtemp+r, ytemp-r, ytemp+r in cursor$
    BOX CLEAR xtemp-r, xtemp+r, ytemp-r, ytemp+r
    BOX SHOW cursor$ at -r, -r
END SUB

SUB move(N, x( ), y( ), q( ), xmax, ymax, dx, dy, r, cursor$, save$, xcursor, ycursor)
    LET xcursor = 0                     ! начальная x-координата курсора
    LET ycursor = 0                     ! начальная y-координата курсора
    DO
      LET choice = 0
      IF KEY input then
        GET KEY choice
        LET xold = xcursor
        LET yold = ycursor
        SELECT CASE choice
        CASE 108                        ! переместить курсор влево
           IF xcursor > -xmax then LET xcursor = xcursor - 2*dx
        CASE 114                        ! переместить курсор вправо
           IF xcursor < xmax then LET xcursor = xcursor + 2*dx
        CASE 117                        ! переместить курсор вверх
           IF ycursor > -ymax then LET ycursor = ycursor + 2*dy
        CASE 100                        ! переместить курсор вниз
```

```
            IF ycursor < ymax then LET ycursor = ycursor - 2*dy
         CASE 112                    ! рисовать силовые линии
            ! перерисовываем силовые линии под курсором
            BOX SHOW save$ at xcursor-r, ycursor-r
            CALL draw(N, x, y, q, xcursor, ycursor)
            BOX KEEP xcursor-r, xcursor+r, ycursor-r, ycursor+r in save$
         CASE ELSE
         END SELECT
         BOX SHOW save$ at xold-r, yold-r
         BOX KEEP xcursor-r, xcursor+r, ycursor-r, ycursor+r in save$
         BOX SHOW cursor$ at xcursor-r, ycursor-r
      END IF
   LOOP until choice = 115        ! стоп
END SUB

SUB draw(N, x(), y(), q(), xcursor, ycursor)
   LET xline = xcursor
   LET yline = ycursor
   LET delta = 0.01
   LET stop_plot = 0
   DO
      LET Ex = 0.0
      LET Ey = 0.0
      FOR i = 1 to N
         LET dx = xline - x(i)  ! расстояние от точки до заряда i по x
         LET dy = yline - y(i)  ! расстояние от точки до заряда i по y
         LET r = sqr(dx*dx + dy*dy)
         IF r > 0.001 then
            LET E0 = q(i)/(r*r*r)
            LET Ex = Ex + E0*dx
            LET Ey = Ey + E0*dy
         ELSE
            LET stop_plot = 1
         END IF
      NEXT i
      LET E = sqr(Ex*Ex + Ey*Ey)
      IF E > 0.001 then
         LET xline = xline + delta*Ex/E ! новое положение силовой линии
         LET yline = yline + delta*Ey/E
      ELSE
```

```
        LET stop_plot = 1
      END IF
      IF stop_plot = 0 then PLOT xline,yline;
    LOOP until key input or stop_plot = 1
    PLOT                                  ! гасим луч
END SUB
```

ЗАДАЧА 9.1. Силовые линии электрического поля системы точечных зарядов

а. В программе **fieldline** не фигурирует кулоновская постоянная K. Почему? Имеет ли в данном случае значение, в каких единицах измерять заряд и расстояние?

б. С помощью программы **fieldline** нарисуйте силовые линии для $q(1) = 1$, $x(1) = 1$, $y(1) = 0$. Обратите внимание, как используются произвольные единицы. Затем добавьте заряд $q(2) = -4$ в точке $x(2) = -1$, $y(2) = 0$. И наконец, добавьте третий заряд $q(3) = 3$ в точке $x(3) = 0$, $y(3) = 1$. Не забудьте, что число силовых линий, исходящих из положительного заряда, выбирается пропорциональным величине данного заряда. Убедитесь, что силовые линии никогда не заканчиваются на положительных зарядах и всегда идут к отрицательным зарядам. Почему силовые линии никогда не пересекаются?

в. Нарисуйте силовые линии электрического диполя $q(1) = 1$, $x(1) = 1$, $y(1) = 0$ и $q(2) = -1$, $x(2) = -1$, $y(2) = 0$.

г. Нарисуйте силовые линии электрического квадруполя $q(1) = 1$, $x(1) = 1$, $y(1) = 1$, $q(2) = -1$, $x(2) = -1$, $y(2) = 0$, $q(3) = 1$, $x(3) = -1$, $y(3) = -1$ и $q(4) = -1$, $x(4) = 1$, $y(4) = -1$.

ЗАДАЧА 9.2. Силовые линии электрического поля квазинепрерывных распределений заряда

а. Непрерывное распределение заряда можно представить большим числом близко расположенных точечных зарядов. Нарисуйте силовые линии электрического поля, создаваемого цепочкой из десяти равноотстоящих единичных зарядов, расположенных на оси x от -5 до 5. Как соотносится это распределение электрического поля с распределением, обусловленным точечным зарядом?

б. Повторите п. «а» с двумя рядами равноотстоящих положительных зарядов, расположенных при $y = 0$ и $y = 1$ соответственно.

в. Повторите п. «б» с одним рядом положительных зарядов и одним рядом отрицательных.

ЗАДАЧА 9.3. Движение заряженной частицы в электрическом поле

а. Добавьте к программе **fieldline** подпрограмму расчета движения частицы с зарядом q в электрическом поле, создаваемом распределением неподвижных точечных зарядов. Для нахождения координаты и скорости частицы используйте алгоритм Верле в скоростной форме (см. приложение 5А или гл. 6). Масса заряда равна m, так что ускорение заряда составляет $(q/m)\mathbf{E}$, где $\mathbf{E}$ — электрическое поле, обусловленное неподвижными точечными зарядами. Мы измеряем заряд в единицах мкКл (10^{-6} Кл) и расстояние в сантиметрах. В этих единицах постоянная K в формуле (9.3) становится равной $K = 90\ \text{Н} \cdot \text{см}^2/\text{мкКл}^2$.

б. Предположим, что E создается неподвижным зарядом $q(1) = 1$, расположенным в начале координат. Промоделируйте движение заряженной частицы с массой $m = 1$ г и зарядом $q = 0.1$, находящейся в начальный момент в точке $x = 1$, $y = 0$. Рассмотрите следующие начальные условия для ее скорости: (i) $v_x = 0$, $v_y = 0$; (ii) $v_x = 1$, $v_y = 0$; (iii) $v_x = 0$, $v_y = 1$ и (iv) $v_x = -1$, $v_y = 0$. (Не забудьте про единицы заряда и расстояния.) Нарисуйте силовые линии, начинающиеся около исходной точки (x,y), и начертите линии, изображающие траектории частиц. Почему траектория частицы не идет по силовой линии?

в. Предположите, что электрическое поле создается точечным зарядом $q(1) = 1$ в начале координат, и определите траекторию частицы для $x_0 = 1$, $y_0 = 0$, $v_{x0} = 0$ и $v_{y0} = 9.5$. Какой вид имеет траектория в этом случае?

г. Предположите, что имеются два неподвижных точечных заряда $q(1) = 1$, $x(1) = 2$, $y(1) = 0$ и $q(2) = -1$, $x(2) = -2$, $y(2) = 0$. Поместите в точку $x_0 = 0.05$, $y_0 = 0$ заряд $q = 1$. Как, по вашему мнению, будет двигаться заряд? Проведите моделирование и определите качественно характер движения.

*д. Рассмотрите движение частицы в окрестности электрического диполя. Сумеете ли вы найти сколько-нибудь ограниченных орбит?

В гл. 4 мы много занимались моделированием движения тел под действием гравитационных сил. Поскольку гравитационное взаимодействие является притягивающим, мы делали упор на изучение ограниченных орбит. В задаче 9.4 будет рассмотрено *рассеяние* положительно заряженной частицы на положительном ядре.

ЗАДАЧА 9.4. Рассеяние альфа-частиц

а. Используя алгоритм Верле в скоростной форме (см. приложение 5А или гл. 6), напишите программу расчета траектории альфа-частицы в окрестности ядра атома золота (ядро атома мишени в первом эксперименте Резерфорда по рассеянию). Соответствующие параметры равны: $m_\alpha = 6.65 \cdot 10^{-27}$ кг, $m_{Au} = 3.27 \cdot 10^{-25}$ кг, $q_\alpha = +2e$ и $q_{Au} = +79e$, где $e = 1.60 \cdot 10^{-19}$ Кл. Будем считать, что ядро атома золота неподвижно. В более точном расчете уравнения надо было бы записать в относительных координатах и использовать приведенную массу. Предполагается, что в действительности α-частица не проникает в ядро, так что взаимодействие между частицей и ядром определяется законом Кулона. В таком случае ускорение альфа-частицы пропорционально величине κ, равной

$$\kappa = K(2e)(79e)/m_\alpha \approx 5.60 \text{ м}^3/\text{с}^2. \qquad (9.4)$$

Поскольку характерное расстояние взаимодействия порядка 10^{-13}см, то для силы, выраженной в ньютонах, будут получаться слишком большие числа, которые не годятся для компьютера. Мы измеряем расстояния в ферми (1 ферми = 10^{-15} м), а скорости — в единицах скорости света $c = 3 \cdot 10^8$ м/с. В этих единицах силовая постоянная κ становится равной

$$\kappa \approx 5.60 \cdot (3 \cdot 10^8)^2 (1 \cdot 10^{-15}) = 5.04 \cdot 10^2 \text{ ферми} \cdot c^2. \qquad (9.5)$$

В обычном эксперименте по рассеянию рассеивающаяся частица сначала находится далеко от ядра мишени, там где кулоновская сила пренебрежимо мала, и частица движется в сторону ядра прямолинейно с постоянной скоростью v. После рассеяния частица будет дви-

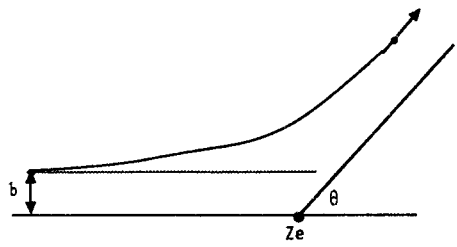

Рис. 9.1. Рассеяние альфа-частицы в зависимости от прицельного параметра b и угла рассеяния θ.

гаться прямолинейно с постоянной скоростью $v' = v$. Рассматриваемыми величинами являются: *прицельный параметр* b, *угол рассеяния* θ (рис. 9.1) и сечение рассеяния, которое связано с вероятностью рассеяния на угол между θ и $\theta + \Delta\theta$. Это сечение представляет собой эффективную площадь акта рассеяния.

б. Возьмите начальные координаты α-частицы равными $x_0 = -300$ ферми, $y_0 = 30$ ферми, $v_{x0} = 0.1\,c$ и $v_{y0} = 0$. Ядро находится в начале координат. Сравните начальное расстояние α-частицы от ядра с радиусом ядра золота, составляющим около 10 ферми. Чему равен прицельный параметр? Рассчитайте траекторию α-частицы и определите угол рассеяния.

в. Определите угол рассеяния для начальных условий $y_0 = 60$ ферми и $y_0 = 120$ ферми. Объясните, почему угол рассеяния является возрастающей или убывающей функцией прицельного параметра.

*г. Возьмите 100 случайных значений прицельного параметра между $b = 0$ и $b = 500$ и подсчитайте, сколько раз $N(\theta)\Delta\theta$ частица рассеивается на угол между θ и $\theta + \Delta\theta$. Выберите подходящее значение $\Delta\theta$ и нарисуйте график $N(\theta)$ как функцию θ. Случайные значения прицельного параметра можно получить с помощью инструкции **rnd**, например LET b = 500*rnd. Объясните качественно зависимость $N(\theta)$.

Известно, что часто легче исследовать свойства системы, рассматривая энергии, а не силы. С этой целью полезно ввести понятие электрического потенциала $V(r)$, определяемого соотношением

$$V(\mathbf{r}_2) - V(\mathbf{r}_1) = -\int_{P_1}^{P_2} \mathbf{E} \cdot d\mathbf{r} \tag{9.6а}$$

или

$$\mathbf{E}(\mathbf{r}) = -\nabla V(\mathbf{r}). \tag{9.6б}$$

Обратите внимание на то, что $V(r)$ является скалярной величиной и что физический смысл имеет только разность потенциалов в двух точках. Оператор градиента ∇ в декартовых координатах определяется формулой

$$\nabla V(\mathbf{r}) = \frac{\partial V(\mathbf{r})}{\partial x}\hat{i} + \frac{\partial V(\mathbf{r})}{\partial y}\hat{j} + \frac{\partial V(\mathbf{r})}{\partial z}\hat{k}. \tag{9.7}$$

Векторы $\hat{i}$, $\hat{j}$ и $\hat{k}$ — это единичные векторы, направленные вдоль осей x, y и z. В одномерном случае выражение (9.6б) сводится к $E = -dV/dx$. Если V зависит только от модуля $\mathbf{r}$, то формула (9.6б) переходит в $E = -dV/dr$. В обоих случаях направление вектора $\mathbf{E}$ совпадает с направлением наиболее быстрого убывания электрического потенциала, что является общим свойством $\mathbf{E}$. Отметим, что потенциал V точечного заряда q при условии, что на бесконечности потенциал равен нулю, имеет вид

$$V(r) = \frac{q}{4\pi\varepsilon_0 r}. \tag{9.8}$$

Поверхность, на которой электрический потенциал принимает везде одинаковые значения, называется *эквипотенциальной поверхностью* (в двумерном случае — кривая). Легко показать, что в любой точке силовые линии электрического поля ортогональны к эквипотенциальным поверхностям. Этим свойством электрических силовых линий и эквипотенциальных линий можно воспользоваться, чтобы с помощью программы **fieldline** нарисовать последние. Поскольку компоненты вектора прямолинейного отрезка Δs электрической силовой линии равны $\Delta x = \Delta s(E_x/E)$ и $\Delta y = \Delta s(E_y/E)$, то компоненты отрезка, перпендикулярного $\mathbf{E}$, а значит, параллельного эквипотенциальной линии, равны $\Delta x = -\Delta s(E_y/E)$ и $\Delta y = \Delta s(E_x/E)$. Неважно, какой знак присвоить компонентам x или y, поскольку это скажется лишь на направлении рисования.

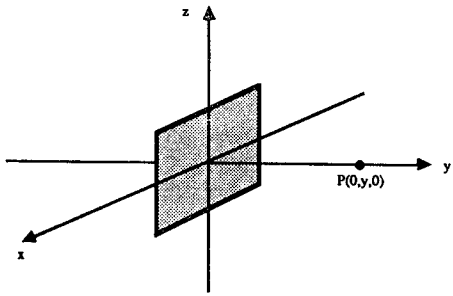

Рис. 9.2. Пластина с полным зарядом Q и площадью L^2 в плоскости x-z.

ЗАДАЧА 9.5. Эквипотенциальные линии

а. Модифицируйте программу **fieldline** и найдите эквипотенциальные линии для распределений заряда, рассмотренных в задаче 9.1.

б. Если бы мы провели такие эквипотенциальные линии, что каждая из них отличалась от соседних на фиксированную разность потенциалов, то что в этом случае означала бы бóльшая и меньшая плотность линий?

в. Объясните, почему эквипотенциальные поверхности никогда не пересекаются.

*ЗАДАЧА 9.6. Электрический потенциал однородно заряженной пластины

Рассмотрим однородно заряженную квадратную пластину со стороной L и полным зарядом Q, лежащую в плоскости x-y (рис. 9.2). Известно, что в пределе $L \longrightarrow \infty$ при постоянной плотности заряда $\sigma = Q/L^2$ электрическое поле нормально к плоскости пластины и равно $E_n = \sigma/2\varepsilon_0$. Наша задача состоит в том, чтобы найти электрическое поле заряженной пластины конечных размеров. Можно действовать следующим образом: пластина разбивается сеткой из $N \times N$ ячеек достаточно мелкой, чтобы каждую ячейку можно было считать точечным зарядом величиной $dq = Q/N^2$. Поскольку потенциал есть величина скалярная, легче рассчитать полный потенциал, а не полное электрическое поле от N^2 точечных зарядов. В свою очередь электрическое поле можно определить с помощью соотношения

(9.6б). Воспользуйтесь выражением (9.8) для потенциала точечного заряда и составьте программу вычисления $V(y)$ и тем самым E_y в точках оси y, перпендикулярной пластине. Сначала проведите расчеты с $L = 1$ см, $Q = 1$ Кл и $N = 10$. Далее увеличивайте N до тех пор, пока результаты для $V(y)$ не перестанут существенно меняться. Начертите графики $V(y)$ и E_y как функции от y и сравните зависимость $V(y)$ и E_y от y с аналогичными характеристиками для бесконечной пластины (на средних расстояниях) и точечного заряда (на больших расстояниях).

9.3. МАГНЕТИЗМ И СИЛОВЫЕ ЛИНИИ МАГНИТНОГО ПОЛЯ

Изучение нами электростатики в разд. 9.2 базировалось на законе Кулона для электрического поля точечного заряда. Аналогичный закон имеется и для магнитного поля, создаваемого током. Это соотношение известно как *закон Био—Савара* и имеет вид

$$\Delta\mathbf{B}(\mathbf{r}) = \frac{\mu_0 I}{4\pi}\left(\frac{\Delta\mathbf{L} \times \mathbf{r}}{r^3}\right), \tag{9.9}$$

где ток I измеряется в амперах (А), магнитное поле B—в теслах (Тл) и магнитная проницаемость μ равна

$$\mu_0 = 4\pi \cdot 10^{-7}\ \text{Тл} \cdot \text{м/А}. \tag{9.10}$$

$\Delta\mathbf{B}(\mathbf{r})$ есть магнитное поле в точке $\mathbf{r}$, создаваемое находящимся в начале координат участком провода $\Delta\mathbf{L}$, по которому протекает постоянный ток I. Закон Био—Савара имеет совершенно общий характер и в принципе может быть использован для нахождения полного магнитного поля в любой точке. Разумеется, никаких изолированных участков тока не бывает, и провод должен либо образовывать замкнутую петлю, либо быть достаточно длинным, чтобы влияние концов было пренебрежимо мало. Однако мы можем приближенно представить непрерывный провод в виде ряда дискретных участков. Вклад каждого участка длиной $\Delta\mathbf{L}$ в точке $\mathbf{r}_j$ в магнитное поле в точке $\mathbf{r}$ определяется выражением (рис. 9.3):

$$\begin{aligned}
\Delta B_x(\mathbf{r}) &= (\mu_0 I/4\pi)[\Delta L_y(z - z_j) - \Delta L_z(y - y_j)]/|\mathbf{r} - \mathbf{r}_j|^3, \\
\Delta B_y(\mathbf{r}) &= (\mu_0 I/4\pi)[\Delta L_z(x - x_j) - \Delta L_x(z - z_j)]/|\mathbf{r} - \mathbf{r}_j|^3, \\
\Delta B_z(\mathbf{r}) &= (\mu_0 I/4\pi)[\Delta L_x(y - y_j) - \Delta L_y(x - x_j)]/|\mathbf{r} - \mathbf{r}_j|^3,
\end{aligned} \tag{9.11}$$

где

$$|\mathbf{r} - \mathbf{r}_j|^3 = [(x - x_j)^2 + (y - y_j)^2 + (z - z_j)^2]^{3/2}. \quad (9.12)$$

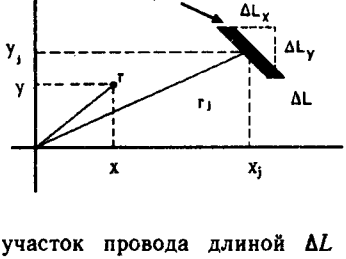

Рис. 9.3. Двумерный участок провода длиной ΔL в точке x_j, y_j. Компоненты $\Delta\mathbf{L}$ равны ΔL_x и ΔL_y. В точке $\mathbf{r}$ вычисляется в данном случае магнитное поле, а в точке $\mathbf{r}_j$ расположен участок провода с током.

Невозможность вычислить магнитное поле аналитически, используя (9.11), если только оно не обладает высокой степенью симметрии, побуждает нас обратиться к помощи компьютера для расчета магнитного поля, создаваемого произвольными конфигурациями проводов с током. Наиболее важными геометрическими конфигурациями проводов, для которых желательно знать распределение магнитного поля, являются прямой провод, петля и катушка (соленоид). Здесь мы рассмотрим петлю с током. Магнитное поле на оси, проходящей через центр петли с током, можно вычислить без особых трудностей аналитически, используя закон Био—Савара. Однако вычислить поле вне этой оси очень трудно. В программе **magnetism** рассматривается круговая петля с током в плоскости *x-z* и рисуются силовые линии магнитного поля в плоскости *x-y* (рис. 9.4). Структура данной программы аналогична структуре программы **fieldline**. Обратите внимание на то, что компоненты *N* участков провода содержатся в массивах.

```
PROGRAM magnetism
DIM x(50), y(50), z(50), dLx(50), dLy(50), dLz(50)
CALL wire(N, a, delta, x, y, z, dLx, dLy, dLz)
CALL screen(a, xmax, ymax, dx, dy, r, cursor$, save$)
CALL move(N, delta, x, y, z, dLx, dLy, dLz, xmax, ymax, dx, dy, r, cursor$, save$)
END
```

```
SUB wire(N, a, delta, x(), y(), z(), dLx(), dLy(), dLz())
  ! вводим участки провода с током
  ! считаем ток равным одному амперу
  INPUT prompt "радиус витка = ": a
  INPUT prompt "число участков = ": N
  INPUT prompt "delta = ": delta ! длина участка магнитной силовой линии
  LET delta_angle = 2*pi/N
  LET angle = 0
  LET dL = 2*pi*a/N
  FOR i = 1 to N
     LET x(i) = a*cos(angle)
     LET y(i) = 0
     LET z(i) = a*sin(angle)
     LET dLx(i) = -dL*sin(angle) ! направление участка тока в точке (x, y, z)
     LET dLy(i) = 0
     LET dLz(i) = dL*cos(angle)
     LET angle = angle + delta_angle
  NEXT i
END SUB

SUB screen(a, xmax, ymax, dx, dy, r, cursor$, save$)
  ! рисуется проекция петли с током на ось x-y
  LET ymax = 3*a
  LET aspect_ratio = 1.5
  LET xmax = aspect_ratio*ymax
  SET window -xmax, xmax, -ymax, ymax
  ! dx и dy – минимальные размеры шагов
  LET dx = 0.025*xmax
  LET dy = 0.025*ymax
  LET r = 0.01*xmax
  LET xtemp = 0.9*xmax
  LET ytemp = 0.9*ymax
  BOX KEEP xtemp-r, xtemp+r, ytemp-r, ytemp+r in save$
  BOX AREA xtemp-r, xtemp+r, ytemp-r, ytemp+r
  BOX KEEP xtemp-r, xtemp+r, ytemp-r, ytemp+r in cursor$
  BOX CLEAR xtemp-r, xtemp+r, ytemp-r, ytemp+r
  BOX SHOW cursor$ at -r, -r
  BOX CIRCLE -a, a, -0.4, 0.4
END SUB
```

```
SUB move(N, delta, x(), y(), z(), dLx(), dLy(), dLz(), xmax, ymax, dx, dy, r, cursor$, save$)
  LET xcursor = 0
  LET ycursor = 0
  DO
    LET choice = 0
    IF KEY input then
      GET KEY choice
      LET xold = xcursor
      LET yold = ycursor
      SELECT CASE choice
      CASE 108                    ! ascii-код литеры "l"
        IF xcursor > -xmax then LET xcursor = xcursor - 2*dx
      CASE 114                    ! ascii-код литеры "r"
        IF xcursor < xmax then LET xcursor = xcursor + 2*dx
      CASE 117                    ! ascii-код литеры "u"
        IF ycursor > -ymax then LET ycursor = ycursor + 2*dy
      CASE 100                    ! ascii-код литеры "d"
        IF ycursor < ymax then LET ycursor = ycursor - 2*dy
      CASE 112                    ! ascii-код литеры "p"
        BOX SHOW save$ at xcursor-r, ycursor-r
        CALL draw(N, delta, x, y, z, dLx, dLy, dLz, xcursor, ycursor)
        BOX KEEP xcursor-r, xcursor+r, ycursor-r, ycursor+r in save$
      CASE ELSE
      END SELECT
      BOX SHOW save$ at xold-r, yold-r
      BOX KEEP xcursor-r, xcursor+r, ycursor-r, ycursor+r in save$
      BOX SHOW cursor$ at xcursor-r, ycursor-r
    END IF
  LOOP until choice = 115         ! ascii-код литеры "s"
END SUB
```

```
SUB draw(N, delta, x(), y(), z(), dLx(), dLy(), dLz(), xcursor, ycursor)
  LET rx = xcursor
  LET ry = ycursor
  DO
    LET Bx = 0.0
    LET By = 0.0
    LET Bz = 0.0
    FOR i = 1 to N
      ! вычисляем расстояние от данной точки до участка тока
      LET dx = rx - x(i)
      LET dy = ry - y(i)
      LET dz = zcursor - z(i)
      LET r = sqr(dx*dx + dy*dy + dz*dz)
      LET B0 = 1/(r*r*r)      ! ток считаем равным 1 A
      ! B пропорционально dL x r
      LET Bx = Bx + B0*(dLy(i)*dz - dLz(i)*dy)
      LET By = By + B0*(dLz(i)*dx - dLx(i)*dz)
      LET Bz = Bz + B0*(dLx(i)*dy - dLy(i)*dx)
    NEXT i
    LET B = sqr(Bx*Bx + By*By + Bz*Bz)
    LET rx = rx + delta*Bx/B      ! новая точка на силовой линии
    LET ry = ry + delta*By/B
    LET rz = rz + delta*Bz/B
    PLOT rx, ry;
  LOOP until key input
  PLOT
END SUB
```

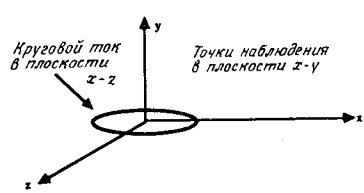

Рис. 9.4. Геометрия кругового тока к задаче 9.7.

ЗАДАЧА 9.7. Магнитное поле петли с током

а. С помощью программы **magnetism** определите силовые линии магнитного поля круговой петли с током, лежащей в плоскости *x-z* с центром в начале координат (см. рис. 9.4). Мы располагаем петлю в плоскости *x-z*, поскольку требуется, чтобы в качестве экрана выступала плоскость *x-y*. Интересующие нас силовые линии лежат в плоскости, перпендикулярной плоскости витка. Задайте радиус петли *a* равным 5 см и *delta* — длину прямолинейного участка магнитных силовых линий — равной 0.1 см. Имеет ли в данном случае значение величина силы тока? Опишите характер магнитных силовых линий. Поле вдали от витка называют дипольным магнитным полем, а сам виток — диполем.

б. Возможно, вы обнаружите, что, когда начинаете магнитную силовую линию вблизи витка тока, эта линия круто загибается и не образует замкнутой петли. Как должно быть на самом деле? Каким образом можно улучшить результаты для таких магнитных силовых линий?

*ЗАДАЧА 9.8. Движение заряженных частиц в магнитных линзах

а. Рассмотрим движение заряженной частицы в магнитном поле, создаваемом коротким соленоидом, который представляет собой цилиндрическую катушку провода (рис. 9.5). Предположим, что провод намотан плотно и с постоянным шагом, причем на единицу длины намотки приходится *n* витков. Несмотря на то что в действительности ток движется по спирали, мы можем считать соленоид эквивалентным

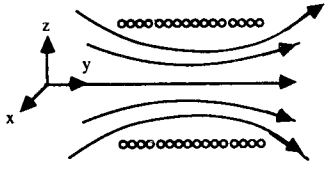

Рис. 9.5. Геометрия короткого соленоида, рассматриваемого в задаче 9.8.

пачке токовых колец. (Поле внутри соленоида не зависит от спиральности катушки.) Напишите подпрограмму вычисления магнитного поля внутри катушки и вблизи нее. Положите I = 1.0 А, радиус a = = 0.5 см, длину катушки L = 1.0 см и количество витков n = 10.

б. Предположим, что электрон влетает в соленоид слева с начальными условиями $z = 0$, $v_z = 0.1$ см/с, $y = 0.125$ см и $v_y = 0.5$ см/с. Напишите подпрограмму расчета траектории электрона в магнитном поле соленоида. Заметим, что вам не надо находить сначала магнитное поле везде. Вместо этого определяйте магнитное поле только в точках нахождения заряда по ходу его движения. Опишите движение электронов, падающих на соленоид в направлении, примерно параллельном оси. Рассмотрите малые изменения в начальных значениях y и объясните, каким образом соленоид действует в качестве линзы.

9.4. ЧИСЛЕННОЕ РЕШЕНИЕ УРАВНЕНИЯ ЛАПЛАСА

В разд. 9.1 мы находили суммарное электрическое поле и электрический потенциал, обусловленные стационарным распределением заряженных источников. Во многих случаях местоположение исходных зарядов нам неизвестно, а известен электрический потенциал на границах области. Предположим, например, что имеется система неподвижных проводников, помещенных в вакуум, и что каждый проводник подсоединен к батарее. Не составляет труда провести измерения и определить потенциал V каждого проводника. (Напомним, что в проводящем теле потенциал V имеет везде одно и то же значение.) Однако измерить положение зарядов на каждом проводнике непросто, поскольку их местоположение определяется сложным неоднородным распределением, которое зависит от формы тела.

Пусть задан потенциал на какой-то системе границ и требуется найти потенциал $V(r)$ в любой точке области, где нет зарядов. Как только мы узнаем V внутри области, для нахождения $\mathbf{E}$ можно воспользоваться соотношением $\mathbf{E} = -\nabla V(\mathbf{r})$. Такая задача называется *краевой*. Прямой метод нахождения $V(x,y,z)$ основан на уравнении Лапласа, которое в декартовых координатах имеет вид

$$\nabla^2 V(x,y,z) \equiv \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} = 0. \qquad (9.13)$$

В данном прямом методе задача заключается в том, чтобы найти функцию $V(x,y,z)$, которая удовлетворяет уравнению (9.13) и, кроме того, удовлетворяет заданным краевым условиям. В силу отсутствия каких бы то ни было аналитических методов для проводников произвольной формы единственным общим подходом является использование приближенных численных методов.

Уравнение Лапласа не есть какой-то новый физический закон, а может быть получено из закона Гаусса. Поскольку вывод уравнения Лапласа дается во всех вводных курсах (см. книги Р. Айсберга и Л. Лернера или Э. П рселла), мы с помощью программы **fieldline** проверим правильность разностной формы уравнения Лапласа, широко применяемой в численном анализе. Все пространство разобьем сеткой или решеткой на мелкие квадратные ячейки (в трехмерном случае — кубические ячейки). Ниже мы покажем, что в отсутствие заряда в точке (x,y) потенциал $V(x,y)$ определяется в двумерном случае уравнением

$$V(x,y) \approx \frac{1}{4}[V(x+\Delta x,y) + V(x-\Delta x,y) + V(x,y+\Delta y) + V(x,y-\Delta y)]. \qquad (9.14)$$

Иначе говоря, $V(x,y)$ равняется среднему по соседним ячейкам справа, слева, сверху и снизу (в трехмерном случае — шесть соседних ячеек). Это замечательное свойство $V(x,y)$ есть не что иное, как дискретный аналог уравнения Лапласа. Приближенную формулу (9.14) можно также подтвердить, аппроксимируя частные производные в уравнении (9.13) конечными разностями.

Чтобы убедиться в правильности формулы (9.14), воспользуемся выражением (9.8) для потенциала $V(r)$ точечного заряда q. Выражение (9.8) есть следствие закона Гаусса и удовлетворяет уравнению Лапласа при $r \neq 0$. Согласно нашему плану, надо модифицировать программу **fieldline** таким образом, чтобы она выводила величину потенциала в точке нахождения курсора. В задаче 9.9 мы выбираем ячейку, вычисляем в ней потенциал и затем перемещаем курсор в четыре соседние ячейки. Если формула (9.14) верна, то среднее значение потенциала по четырем соседним ячейкам должно равняться потенциалу центральной ячейки.

Следующая подпрограмма будет использована в задаче 9.9. Подпрограмма **potential** выводит значение потенциала в точке нахождения курсора.

```
SUB potential(N, x(), y(), q(), xcursor, ycursor)
   FOR i = 1 to N
      LET dx = xcursor - x(i)
      LET dy = ycursor - y(i)
      LET r = sqr(dx*dx + dy*dy)
      LET V = V + q(i)/r
   NEXT i
   PRINT "x, y, V = ", xcursor, ycursor, V
END SUB
```

ЗАДАЧА 9.9. Проверка двумерного разностного уравнения для потенциала

а. Для вычисления потенциала в любой требуемой точке замените подпрограмму **draw** в программе **fieldline** на подпрограмму **potential**. Примите, что источником потенциала является точечный заряд 1 мкКл в начале координат. Выберите подходящие значения Δx и Δy и рассмотрите произвольную точку на экране, не слишком близкую к началу координат. Выведите значение потенциала в точке положения курсора, для чего нажмите клавишу 'p'. Переместите курсор в каждую из четырех соседних ячеек и выведите значение V для каждой из них. Вычислите среднее V по четырем соседним ячейкам и сравните полученное значение с величиной V в центральной ячейке. Проделайте аналогичные измерения для других положений курсора. Зависит ли относительное расхождение с формулой (9.14) от расстояния ячейки до начала координат?

б. Повторите п. «а» с меньшими значениями Δx и Δy. Насколько существенно, одинаковы ли Δx и Δy? Свой ответ обоснуйте. Лучше ли согласуются полученные результаты с (9.14)?

в. Повторите п. «а» с двумя точечными зарядами по 1 мкКл каждый, закрепленными в точках $x(1) = 10$ см и $x(2) = -10$ см с $y(1) = y(2) = 0$.

Теперь, когда мы убедились, что равенство (9.14) согласуется с законом Кулона, примем формулу (9.14) за основу вычислительного метода решения задач, где мы не можем определить потенциал непосредственно из закона Кулона. В частности, рассмотрим задачи, в которых несколько проводящих областей имеют некоторый заданный потенциал и

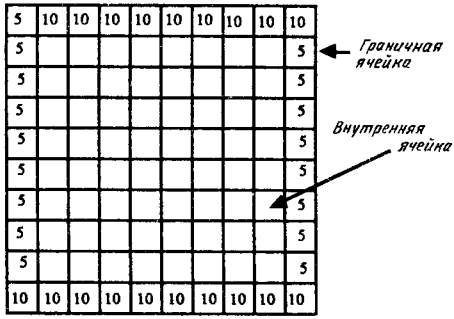

Рис. 9.6. Распределение потенциала для задачи 9.10в. Обратите внимание на то, что в двух из четырех углов потенциал неоднозначен.

требуется найти потенциал во всем остальном пространстве. Для простоты будем рассматривать только двумерные геометрии. Подход, называемый *методом релаксации*, базируется на следующем алгоритме:

1. Разбиваем рассматриваемую область сеткой или системой ячеек, покрывающей всю область (рис. 9.6). Область должна окаймляться поверхностью (в двумерном случае — кривой) с заданным значением потенциала по всей кривой.
2. Ячейки делятся на граничные и внутренние. Присваиваем каждой граничной ячейке, т.е. ячейке, центр которой лежит внутри области с заданным потенциалом, значение потенциала этой области.
3. Присваиваем всем внутренним ячейкам произвольный потенциал (лучше какое-нибудь разумное начальное приближение).
4. На первом шаге для всех внутренних ячеек вычисляем новые значения V. Каждое новое значение получается путем усреднения начальных значений потенциала по четырем ближайшим соседним ячейкам. Это — первая итерация процесса релаксации.
5. Повторяем описанную в п. 4 процедуру, используя значения V, полученные на предыдущей итерации. Данный итерационный процесс продолжается до тех пор, пока потенциал каждой внутренней ячейки не будет меняться в пределах требуемой степени точности.

В задачах 9.10 и 9.11 для расчета потенциала в различных геометриях мы применяем программу **Laplace**, в которой реализован описанный алгоритм.

```
PROGRAM Laplace
DIM V(100,100)
CALL assign(V, nx, nγ, min_change)
CALL iterate(V, nx, nγ, min_change, iterations)
END

SUB assign(V(,), nx, nγ, min_change)
   INPUT prompt "число ячеек по оси x = ": nx
   INPUT prompt "число ячеек по оси γ = ": nγ
   INPUT prompt "потенциал на границе прямоугольника в вольтах = ": V0
   INPUT prompt "относительная точность = ": min_change
   LET min_change = min_change/100
   FOR col = 1 to nx                    ! фиксируем потенциал на границе
      LET V(col,1) = V0
      LET V(col,nγ) = V0
   NEXT col
   FOR row = 1 to nγ
      LET V(1,row) = V0
      LET V(nx,row) = V0
   NEXT row
   ! задаем начальные потенциалы внутренних ячеек
   FOR col = 2 to nx - 1
      FOR row = 2 to nγ - 1
         LET V(col,row) = 0.9*V0
      NEXT row
   NEXT col
END SUB
```

```
SUB iterate(V(,),nx,ny,min_change,iterations)
  DIM Vave(100,100)
  LET iterations = 0
  DO
    LET dmax = 0
    LET iterations = iterations + 1
    FOR col = 2 to nx - 1
      FOR row = 2 to ny - 1
          ! вычисляем средний потенциал соседних ячеек
          LET Vave(col,row) = V(col+1,row) + V(col-1,row)
          LET Vave(col,row) = Vave(col,row) + V(col,row+1) + V(col,row-1)
          LET Vave(col,row) = 0.25*Vave(col,row)
          ! вычисляем относительное изменение потенциала
          LET diff = abs((V(col,row) - Vave(col,row))/Vave(col,row))
          IF diff > dmax then LET dmax = diff
      NEXT row
    NEXT col
    FOR col = 2 to nx - 1        ! изменяем потенциал в каждой ячейке
      FOR row = 2 to ny - 1
          LET V(col,row) = Vave(col,row)
      NEXT row
    NEXT col
    CALL output(V,nx,ny,iterations)
  LOOP until dmax <= min_change  ! повторяем до требуемой точности
END SUB

SUB output(V(,),nx,ny,iterations)              ! вывод результатов
  PRINT
  PRINT "итерация = ", iterations
  FOR row = 1 to ny
    FOR col = nx to 1 step -1
      PRINT using "###.###": V(col,row);
    NEXT col
    PRINT
  NEXT row
END SUB
```

ЗАДАЧА 9.10. Численное нахождение потенциала в прямоугольной области

а. С помощью программы **Laplace** определите потенциал $V(x,y)$ внутри квадратной области с линейным размером L = 10 см. Потенциал на границе квадрата равен 10 В. Прежде чем производить вычисления, угадайте точный вид $V(x,y)$ и задайте начальные значения потенциала внутренних ячеек с точностью 5% от точного ответа. Площадь каждой ячейки примите равной 1 см^2. Сколько итераций необходимо сделать для достижения точности 1%?

б. Рассмотрите ту же геометрию, что в п. «а», но примите начальный потенциал во всех внутренних ячейках равным нулю, кроме точки $V(5,5)$ = 4. Опишите временну́ю эволюцию распределения потенциала внутренних ячеек. Эволюционирует ли распределение потенциала к правильному решению? Зависят ли конечные результаты от выбранного начального приближения? В чем проявляется плохое начальное приближение? Заметим, что в том виде, как написана программа, $V(x,y)$ не может равняться нулю ни для какой внутренней ячейки. (Характер релаксации сеточных значений к конечному равновесному распределению очень близок к процессу *диффузии*, который рассматривается в гл. 11.)

в. Модифицируйте подпрограмму **assign** в программе **Laplace** так, чтобы потенциалы каждой стороны прямоугольника могли быть разными. Используйте по 10 ячеек с каждой стороны и задайте потенциалы сторон равными 5, 10, 5 и 10 В соответственно (см. рис. 9.6). Изобразите схематически эквипотенциальные поверхности. Что будет, если на трех сторонах потенциал равен 10 В, а на четвертой — нулю? Начните с подходящего приближения для начальных значений потенциала внутренних ячеек и итерируйте до получения точности 1%.

г. Повторите п. «в», взяв 20 ячеек с каждой стороны. Насколько улучшаются результаты при увеличении числа ячеек?

д. Мы рассмотрели только очень простую реализацию метода релаксации. Использование более общих релаксационных методов и характер их сходимости рассмотрены во многих учебниках (см. книгу С. Коонина). Как изменятся ваши результаты, если вычислять потенциал в каждом узле последовательно, а не одновременно?

ЗАДАЧА 9.11. Емкость концентрических квадратов

а. Модифицируйте программу **Laplace** так, чтобы рассмотреть границу в виде концентрических квадратов (рис. 9.7). Потенциал внешнего квадратного проводника равен 10 В, а внутреннего квадратного проводника, который расположен в центре внешнего квадрата, равен 5 В. Линейные размеры наружного и внутреннего квадратов составляют L_1 = 5 см и L_2 = 25 см соответственно. Выберите подходящую сетку и вычислите распределение потенциала между этими двумя квадратами. Изобразите схематически эквипотенциальные поверхности. Не забудьте изменить программу так, чтобы потенциал внутреннего квадрата оставался неизменным.

б. Емкость C системы двух проводников с зарядами Q и $-Q$ соответственно определяется как отношение Q к разности потенциалов между этими двумя проводниками. Определите емкость концентрических квадратных проводников, рассмотренных в п. «а». В данном случае разность потенциалов равна 5 В. Заряд Q можно вычислить, исходя из того свойства, что вблизи проводящей поверхности плотность поверхностного заряда равна $\sigma = E_n/\varepsilon_0$. Здесь E_n представляет собой абсолютную величину нормальной к поверхности компоненты электрического поля и может быть аппроксимирована выражением $-\Delta V/\Delta r$, где ΔV — разность потенциалов между граничной ячейкой и соседней с ней внутренней ячейкой, находящейся на расстоянии Δr. (Мы считаем, что оба квадрата в третьем измерении бесконечны.)

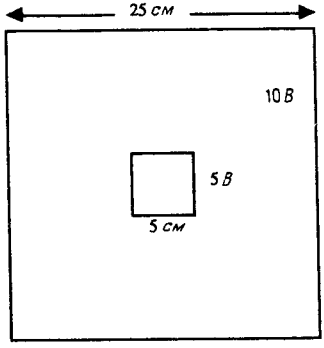

Рис. 9.7. Расположение двух концентрических квадратов, рассматриваемых в задаче 9.11.

Воспользуйтесь результатами п. «а» и вычислите ΔV для всех узлов, примыкающих к обеим квадратным поверхностям. С помощью этой информации получите разумную оценку для E_n на обеих поверхностях и линейной плотности заряда на каждом электроде. Одинакова ли величина этих зарядов и разные ли у них знаки? Подсчитайте емкость в фарадах. Как соотносится величина этой емкости с емкостью системы двух концентрических цилиндров с радиусами $2\pi r_1 = 4L_1$ и $2\pi r_2 = 4L_2$?

в. Сместите внутренний квадрат на 1 см от центра и повторите все расчеты п.п. «а» и «б». Как изменились поверхности потенциала? Будет ли какая-нибудь качественная разница, если задать потенциал центрального проводника равным -5 В, а не +5 В?

Уравнение Лапласа справедливо только в областях, где нет зарядов. Если же в области имеются заряды с плотностью $\rho(x,y,z)$, то нужно использовать уравнение *Пуассона*, которое в дифференциальной форме можно записать в виде

$$\nabla^2 V(r) = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} = -\frac{\rho(\mathbf{r})}{\varepsilon_0}. \qquad (9.15)$$

Величина $\rho(\mathbf{r})$ есть плотность заряда. В задаче 9.12 мы проводим измерения V в окрестности ячейки с центром в точке (x,y), содержащей исходный заряд. Результат этих измерений согласуется с разностной формой уравнения Пуассона, которая в двумерном случае имеет вид

$$V(x,y) \approx \tfrac{1}{4}\,[\,V(x+\Delta x,y) + V(x-\Delta x,y) + V(x,y+\Delta y) + V(x,y-\Delta y)] + \\ + \tfrac{1}{4}\Delta x\Delta y\,\frac{\rho(x,y)}{\varepsilon_0}. \qquad (9.16)$$

Заметим, что $\rho(x,y)\Delta x\Delta y$ есть просто полный заряд в ячейке с центром в точке x,y. Величина ε_0 связана с кулоновской постоянной соотношением $K = 1/4\pi\varepsilon_0$ и равна $8.85\cdot10^{-12}$ Кл$^2\cdot$Н$^{-1}\cdot$м$^{-2}$.

ЗАДАЧА 9.12. Разностная форма уравнения Пуассона

а. Поместите одиночный заряд 1.0 мкКл в ячейку (0,0), отмеченную на рис. 9.8 буквой *с*. В этом случае нельзя прямо вычислить потенциал ячейки *с*, поскольку в ней находится точечный заряд. Ис-

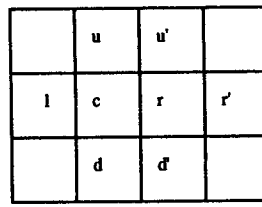

Рис. 9.8. Единичный точечный заряд находится в ячейке *c*.

пользуя закон Кулона и программу **fieldline**, найдите приближенно потенциал в ближайших соседних с *c* ячейках (помечены буквами *u*, *d*, *l* и *r*). С помощью этих результатов получите средний потенциал $\langle V_c \rangle$.

б. Найдите средний потенциал $\langle V_r \rangle$ ячейки *r* путем усреднения по ее ближайшим соседям *u'*, *r'*, *d'* и *c*. Возьмите потенциал ячейки *c* равным $\langle V_c \rangle$, т.е. среднему, найденному в п. «а». Затем вычислите потенциал V_r ячейки *r* непосредственно и подсчитайте разность между $\langle V_r \rangle$ и V_r.

в. Повторите вычисления п.п. «а» и «б» для точечного заряда 2 мкКл в начале координат. Повторите эти вычисления для точечного заряда −4 мкКл в начале координат. Покажите, что разность между $\langle V_r \rangle$ и V_r пропорциональна заряду в точке (0, 0). Вид этой разности указывает на то, что потенциал в любой ячейке приближенно равен среднему потенциалу, найденному по ее ближайшим соседям, плюс член, пропорциональный заряду в данной ячейке. В принципе коэффициент пропорциональности можно было бы найти, усредняя потенциал точечного заряда по конечной пространственной области данной ячейки.

ЗАДАЧА 9.13. Численное решение уравнения Пуассона

а. Рассмотрите квадрат со стороной L = 25 см, находящийся под заданным потенциалом 10 В. Предположите, что в каждой внутренней ячейке заряд распределен однородно с плотностью ρ, такой что $\frac{1}{4} A\rho/\varepsilon_0$ = 3 В/см^2. Модифицируйте программу **Laplace**, чтобы вычис-

лять распределение потенциала для этого случая. Сравните эквипотенциальные поверхности, полученные для этого случая, с найденными в задаче 9.11.

6. Найдите распределение потенциала, если распределение заряда в п. «а» ограничено квадратом площадью 25 $см^2$ в центре.

9.5. ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ

Несмотря на то что мы иллюстрировали метод релаксации для уравнений Лапласа и Пуассона на примерах задач электростатики, аналогичные уравнения применяются в магнитостатике для компонент векторного потенциала. Кроме того, столь разные объекты, как установившийся тепловой поток, статические прогибы упругих мембран и безвихревое течение жидкости, также описываются аналогичными уравнениями.

ЛИТЕРАТУРА

Forman S. Acton, Numerical Methods That Work, Harper & Row, 1970. В гл. 18 рассматривается решение уравнения Лапласа методом релаксации и другими методами.

Charles K. Birdsall, A. Bruce Langdon, Plasma Physics via Computer Simulation, McGraw-Hill, 1985. [Имеется перевод: *Ч. Бэрдсол, А. Ленгдон*, Физика плазмы и численное моделирование. — М.: Энергоатомиздат, 1989.]

David M. Cook, The Theory of the Electromagnetic Field, Prentice-Hall, 1975. Одна из первых книг, в которой численные методы вводятся в контексте электромагнетизма.

J. M. Dawson, Particle simulations of plasma, Rev. Mod. Phys. **55**, 403 (1983).

Robert M. Eisberg, Lawrence S. Lerner, Physics, Vol. 2, McGraw-Hill, 1981. Начальный курс, в котором численные методы используются для нахождения силовых линий электрического поля и решения уравнения Лапласа.

R. H. Good, Dipole radiation: Simulation using a microcomputer, Am. J. Phys. **52**, 1150 (1984). Автор описывает графическое моделирование дипольного излучения.

R. W. Hockney, J. W. Eastwood, Computer Simulation Using Partic-

les, McGraw-Hill, 1981. [Имеется перевод: *Р. Хокни, Дж. Иствуд*, Численное моделирование методом частиц. — М.: Мир, 1987.]

Steven E. Koonin, Computational Physics, Benjamin/Cummings, 1986. [Готовится перевод: *С. Кунин*, Вычислительная физика. — М.: Мир, 1991.] См. гл. 6, где говорится о численном решении эллиптических уравнений в частных производных, частным случаем которых являются уравнения Лапласа и Пуассона.

Edward M. Purcell, Electricity and Magnetism, 2nd ed., Berkely Physics Course, Vol. 2, McGraw-Hill, Inc., 1985. [Имеется перевод: *Э. Перселл*, Электричество и магнетизм. — М.: Мир, 1982.] Широко известный учебник, в котором рассмотрен метод релаксации.

T. Tajima, A. Clark, G. G. Craddock, et al., Particle simulation of plasmas and stellar systems, Am. J. Phys. **53**, 365 (1985). В рамках единого подхода авторы рассматривают моделирование галактик и электронной плазмы.

P. B. Visscher, Fields and Electrodynamics, John Wiley & Sons, (готовится к печати). Книга средней сложности, в которой рассматриваются как численные модели, так и аналитические методы. Особый интерес представляет разработанная авторами дискретная версия электродинамики, которая удовлетворяет закону Гаусса, закону сохранения энергии и др. для произвольного периода кристаллической решетки.

Gregg Williams, An introduction to relaxation methods, Byte **12**, 111 (January, 1987). Автор рассматривает применение релаксационных методов к решению двумерного уравнения Пуассона.

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

Сивухин Д. В., Электричество и магнетизм. — М.: Наука, 1986. В гл. 1—3 подробно рассмотрены вопросы, касающиеся электростатических и магнитных полей.

Кунц К. С. Численный анализ. — Киев: Техніка, 1964. В гл. 13 подробно исследуется решение уравнения Лапласа.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А. КРАТКАЯ СВОДКА ОСНОВНЫХ СИНТАКСИЧЕСКИХ КОНСТРУКЦИЙ ЯЗЫКОВ БЕЙСИК, ФОРТРАН И ПАСКАЛЬ

Приведенное ниже сравнение может использоваться только в качестве справочника. Под языком Бейсик подразумевается версия Microsoft BASIC для IBM PC.

Бейсик	True BASIC	Фортран	Паскаль
Арифметические операции			
+ - * /	те же	те же	те же
2^3	2^3	2**3	-
Описание переменных			
			var
-	-	integer i	i : integer;
-	-	real x	x : real;
любая переменная, оканчивающаяся знаком $		character a*5	a : packed array[1..5] of char;
dim a(5)	dim a(5)	dimension a(5)	a : array[1..5] of real;
Присваивание			
x = 5	let x = 5	x = 5	x := 5;
Логические отношения			
=	=	.eq.	=
<>	<>	.ne.	<>
>	>	.gt.	>
<	<	.lt.	<
>=	>=	.ge.	>=
<=	<=	.le.	<=
and	and	.and.	and
or	or	.or.	or

Бейсик	True BASIC	Фортран	Паскаль
Арифметические функции			
sqr (x)	sqr (x)	sqrt (x)	sqrt (x)
fix (x)	truncate (x,0)	int (x)	trunc (x)
int (x)	int (x)	–	–
abs (x)	та же	та же	та же
sin (x)	та же	та же	та же
cos (x)	та же	та же	та же
tan (x)	та же	та же	та же
exp (x)	та же	та же	та же
log (x)	log (x)	alog (x)	ln (x)
–	log10 (x)	alog10 (x)	–
–	max (x,y)	amax1 (x,y)	–
–	–	max0 (i,j)	–
–	min (x,y)	amin1 (x,y)	–
–	–	min0 (i,j)	–
rnd	rnd	зависит от реализации	

Циклы

Бейсик

```
10   for i = 1 to n
20      x = x + 1
30   next i

10   while i > 10
20      i = i + 1
30   wend

10   while idum = 0
20      i = i + 1
30      if i >= 10 then goto 50
40   wend
50   'имитация цикла repeat—until
```

True BASIC

```
for i = 1 to n
  let x = x + 1
next i

do while i < 10
  let i = i + 1
loop

do
  let i = i + 1
loop until i >= 10
```

Фортран

```
      do 10 i = 1,n
        x = x + 1
10    continue

      do 10 idum = 1,10000
        if(i.ge.10) go to 100
        i = i + 1
10    continue
100   continue

      do 10 idum = 1,10000
        i = i + 1
        if(i.ge.10) go to 100
10    continue
100   continue
```

Паскаль

```
for i := 1 to n begin
      x := x + 1
end;

while i < 10 do
  begin
    i := i + 1
  end;

repeat
  i := i + 1
until i >= 10;
```

Логические инструкции

Бейсик

```
10  if i > 1 then i = 0
20  if i > 1 then i = 0 else i = -1

30  if i > 1 then i = 0: j = 2 else i = -1
```

True BASIC

```
if i > 1 then let i = 0
if i > 1 then
   let i = 0
else
   let i = -1
end if
if i > 1 then
   let i = 0
   let j = 2
else
   let i = -1
end if
```

Фортран

```
if(i .gt. 1) i = 0

if(i .gt. 1) then
    i = 0
else
    i = -1

if(i .gt. 1) then
    i = 0
    j = 2
else
    i = -1
end if
```

Паскаль

```
if i > 1 then i := 0;

if i > 1
    i := 0
else
    i := -1;

if i > 1 then
    begin
        i := 0
        j := 2
    end
else
    i := -1;
```

ПРИЛОЖЕНИЕ Б. ПРИМЕРЫ ИНСТРУКЦИЙ ВВОДА-ВЫВОДА

Инструкции ввода-вывода определяются аппаратурой компьютера и его системой команд. Приведенные ниже программы на языках Microsoft BASIC, True BASIC, Фортран-77 и Паскаль записывают данные в файл test.dat, читают этот файл и выводят его содержимое на экран, а для персональных компьютеров и на принтер, подсоединенный к компьютеру.

```
10  'Программа IO
20  ' пример ввода-вывода на языке Microsoft BASIC для IBM PC
30  x = 5.76
40  open "test.dat" for output as #1
50  print #1,x
60  close #1
70  open "test.dat" for output as #2
80  print #2,x
90  close #2
100 print using "###.##";x              'вывод на экран
110 lprint using "###.##";x             'вывод на принтер
```

```
Program IO
! пример ввода-вывода на языке True BASIC
let x = 5.76 !
open #1: name "test.dat", access output, create new
print #1:x
close #1
open #2: name "test.dat", access input
input #2:x
close #2
print using "###.##":x
open #1: printer
print #1:x                              ! вывод на принтер
close #1
end
```

```
Program IO (output, data);
(* пример ввода-вывода на языке Turbo Pascal для IBM PC *)
var
   x : real;
   data : text;
begin
   x := 5.76
   assign (data, 'test.dat');   (* присваивание переменной имени файла *)
   rewrite (data);                    (* открытие файла для вывода *)
   writeln (data, x:1:2);
   (* не нужно, если файл используется снова в той же программе*)
   close (data);
   reset (data);                      (* открывает файл для ввода *)
   readln (data, x);
   close (data);
   writeln (x:6:2);              (* имя : длина : число десятичных знаков *)
   writeln (lst, x:6:2);          (* вывод на принтер *)
end.
```

```
Program IO (output, input);
(* пример ввода-вывода на языке VAX Pascal *)
var
   x : real;
   data : text;
begin
   x := 5.76
   open (data, 'test.dat');
   rewrite (data);
   writeln (x);
   close (data);
   open ('test.dat', history :=old);
   reset (data);
   readln (x);
   close (data);
   writeln (x:6:2);
end.
```

```
*       Program IO
*       пример ввода-вывода на языке VAX FORTRAN 77
        x = 5.76
        open (1, file='test.dat')
        write (1) x
        close (1)
        open (2, file='test.dat')
        read (2) x
        close (2)
        write (5,10) x
10      format (f6.2)
        stop
        end
```

ПРИЛОЖЕНИЕ В. УКАЗАТЕЛЬ ПРОГРАММ НА ЯЗЫКЕ TRUE BASIC: ЧАСТЬ 1

Приводится список номеров страниц с программами на языке True BASIC из ч. 1 книги.

Указатель программ моделирования на языке TRUE BASIC

Программа	Страница	Описание
cool	36	Закон остывания Ньютона
cooler	37	Усовершенствованная версия программы cool
fall	56	Движение свободно падающего тела
styrofoam	61	Движение свободно падающего тела с учетом сопротивления воздуха
planet	78	Движение планеты
planet2	92	Движение двух взаимодействующих планет
sho	102	Простой гармонический осциллятор
pendula	111	Мультипликация линейного и нелинейного маятников
rc	123	Моделирование RC-цепи
Beeman	140	Алгоритм Бимана для движения частицы в потенциале Морза
md	150	Программа молекулярной динамики для потенциала Леннарда—Джонса
map_table	186	Табулирование итераций стандартного отображения
map_plot	187	График итераций стандартного отображения
map_graph	193	Граф стандартного отображения, получаемый с помощью рекурсии
nonlinear	205	Хаотическое движение нелинейного маятника
oscillators	216	*N* связанных осцилляторов
Fourier	224	Сложение фурье-гармоник
waves	227	Волновое движение
interfere	233	Интерференция и дифракция волн
polarize	238	Поляризация света
Fermat	244	Принцип Ферма
fieldline	254	Силовые линии электрического поля
magnetism	264	Силовые линии магнитного поля
Laplace	273	Решение уравнения Лапласа методом релаксации

Указатель программ, иллюстрирующих синтаксис языка TRUE BASIC

Программа	Страница	Описание
example	31	Структура программы в языке True BASIC
local	33	Свойства локальных переменных
global1	33	Свойства глобальных переменных
global2	34	Пример передачи параметров в подпрограмму
global3	34	Важность порядка параметров подпрограммы
graphics	44	Графические программы
plot	44	Подпрограмма построения осей для выводимых на график функций
do_loop	56	Использование цикла **DO** и инструкции **WHILE**
array	76	Использование массивов
circle	77	Нахождение характеристического отношения монитора
key	85	Использование конструкции **KEY INPUT**
print	109	Прямая выдача на принтер
animation	110	Использование инструкций **BOX KEEP** и **BOX SHOW**

ПРИЛОЖЕНИЕ Г. РАСПЕЧАТКИ ПРОГРАММ НА ЯЗЫКЕ ФОРТРАН. ЧАСТЬ 1

Ниже приводятся распечатки перевода многих программ, приведенных в тексте, с языка True BASIC на Фортран-77. Обычно мы составляли программы на языке True BASIC таким образом, чтобы перевод их на Фортран и Паскаль можно было осуществить непосредственно. Фортранные программы пропускались на машине VAX 11/750.

К сожалению, между графическими инструкциями, используемыми в различных языках программирования, не существует взаимно однозначного соответствия. Среди языков программирования, имеющихся на персональных компьютерах, True BASIC обладает важным достоинством, состоящим в том, что позволяет пользователю устанавливать систему координат экрана, не зависящую от числа пикселов. Кроме того, True BASIC позволяет располагать наибольшее значение вертикальной координаты вверху экрана.

Поскольку язык Фортран разрабатывался в те годы, когда машинная графика еще не получила широкого распространения, то стандартный Фортран не содержит никаких графических инструкций и подпрограмм. В качестве примера использования графического пакета в Фортране в наших программах проиллюстрировано применение PLOT 10 (фирма Tektronix) — распространенного пакета, имеющегося на машине VAX и других компьютерах. На большинстве компьютеров PLOT 10 может также вызываться из Паскаля. Краткое описание графических подпрограмм из пакета PLOT 10, задействованных в наших программах, приведено в табл. Г1.

ТАБЛИЦА Г1. Основные графические подпрограммы из пакета PLOT 10

Подпрограмма	Описание
initt(ibaud)	Инициализация графики со скоростью ibaud (в бодах)
dwindo(xmin,xmax,ymin,ymax)	Задает оконные координаты
movea(x,y)	Помещает перо в точку с (x,y)
pointa(x,y)	Наносит точку с координатами (x,y)
drawa(x2,y2)	Проводит прямую из текущей точки в точку с координатами (x2,y2)
a1out(nchar,iarray)	Выводит первые nchar символов из текстового массива iarray
tsend	Разгружает выходной буфер
finit(ix,iy)	Выход из графической моды, помещение пера в столбец ix и строку iy

ГЛАВА 2

```
*       начало основной программы
        PROGRAM example
        CALL start(y, x, dx, n)
        CALL Euler(y, x, dx, n)
        STOP
        END
*       конец основной программы

        SUBROUTINE start(y, x, dx, n)
*       начальное значение x
        x = 1
*       максимальное значение x
        xmax = 2
*       начальное значение y
        y = 1
*       величина шага
        dx = 0.1
        n = (xmax - x)/dx
        RETURN
        END

        SUBROUTINE Euler(y, x, dx, n)
*       цикл повторяется n раз
        DO 10 i = 1, n
*           наклон в начальной точке отрезка
            slope = 2.0*x
*           вычисление полного изменения на отрезке
            change = slope*dx
*           новое значение y
            y = y + change
*           приращение величины x
            x = x + dx
            WRITE(6, *) x, y
10      CONTINUE
        RETURN
        END
```

```
      PROGRAM cool
*     метод Эйлера для задачи об остывании кофе
      CALL start(t, temp, rmtemp, r, dt, ncalc)
      CALL Euler(t, temp, rmtemp, r, dt, ncalc)
      STOP
      END

      SUBROUTINE start(t, temp, rmtemp, r, dt, ncalc)
*     начальное время
      t = 0.0
*     начальная температура кофе (C)
      temp = 83
*     комнатная температура (C)
      rmtemp = 32
*     коэффициент остывания (1/мин)
      r = 0.1
*     шаг по времени (мин)
      dt = 0.1
*     длительность (мин)
      tmax = 2
*     общее количество шагов
      ncalc = tmax/dt
      RETURN
      END

      SUBROUTINE Euler(t, temp, rmtemp, r, dt, ncalc)
      DO 10 icalc = 1, ncalc
*        начало отрезка
         change = -r*(temp - rmtemp)
         temp = temp + change*dt
*        время
         t = t + dt
         CALL output(t, temp)
10    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE output(t, temp)
*     печать результатов
      WRITE(6,*) t, temp
      RETURN
      END

      PROGRAM cooler
*     модифицированная программа
      CALL start(t, temp, rmtemp, r, dt, ncalc, nprt)
*     печать начальных значений
      CALL output(t, temp)
      DO 100 iprt = 1, nprt
         CALL Euler(t, temp, rmtemp, r, dt, ncalc)
*        печать результатов
         CALL output(t, temp)
100   CONTINUE
      STOP
      END

      SUBROUTINE start(t, temp, rmtemp, r, dt, ncalc, nprt)
*     начальное время
      t = 0.0
*     начальная температра кофе (C)
      temp = 83
*     комнатная температура (C)
      rmtemp = 32
*     коэффициент остывания (1/мин)
      WRITE(6,*) 'коэффициент остывания r = '
      READ(5,*) r
*     шаг по времени (мин)
      WRITE(6,*) 'шаг по времени dt = '
      READ(5,*) dt
*     длительность (мин)
      WRITE(6,*) 'длительность = '
      READ(5,*) tmax
*     интервал (мин) вывода на печать
      prtper = 0.5
*     число обращений к выводу результатов
      nprt = tmax/prtper
*     число итераций между печатями
```

```
      ncalc = prtper/dt
      WRITE(6,17)
17    FORMAT(5x,'время',7x,'температура')
*     пропуск строки
      WRITE(6,*)
      RETURN
      END

      SUBROUTINE Euler(t, temp, rmtemp, r, dt, ncalc)
      DO 10 icalc = 1, ncalc
         change = -r*(temp - rmtemp)
         temp = temp + change*dt
10    CONTINUE
      t = t + dt*ncalc
      RETURN
      END

      SUBROUTINE output(t, temp)
*     печать результатов
      WRITE(6,*) t, temp
      RETURN
      END
```

ГЛАВА 3

```
      PROGRAM fall
*     свободно падающее тело
*     начальные условия и параметры
      CALL start(y, v, t, g, dt, height)
*     печать параметров
      CALL prtpar(dt, ncalc)
*     печать начальных значений
      CALL prttab(y, v, g, t)
*     10000 выбрано условно как пример большого числа
      DO 100 i = 1,10000
*        решение разностного уравнения
         CALL Euler(y, v, accel, t, g, dt, ncalc)
*        печать результатов через ncalc шагов
         CALL prttab(y, v, accel, t)
*        окончание работы программы, если y > height
         IF (y.gt.height) STOP
100   CONTINUE
      STOP
      END

      SUBROUTINE start(y, v, t, g, dt, height)
*     начальный момент времени (с)
      t = 0.0
*     начальное значение координаты (м)
      y = 0.0
*     начальная высота тела над землей
      height = 10.0
*     начальная скорость (м/с)
      v = 0.0
      WRITE(6,*) 'шаг по времени dt = '
      READ(5,*) dt
*     величина ускорения свободного падения
      g = 9.8
      RETURN
      END
```

```
      SUBROUTINE prtpar(dt,ncalc)
      prtper = 0.1
*     число шагов между печатью результатов
      ncalc = prtper/dt
*     заголовок
      WRITE(6,13)
13    FORMAT(5x,'время(с)',8x,'y(m)',7x,'скорость(м/с)',4x,'ускор(м/с*с)'/)
      RETURN
      END

      SUBROUTINE prttab(y,v,accel,t)
      WRITE(6,15) t,y,v,accel
15    FORMAT(4F15.5)
      RETURN
      END

      SUBROUTINE Euler(y,v,accel,t,g,dt,ncalc)
      DO 10 icalc = 1,ncalc
*        скорость в начале интервала
         y = y + v*dt
*        ось y направлена вниз
         accel = g
         v = v + accel*dt
10    CONTINUE
      t = t + dt*ncalc
      RETURN
      END
```

```
      PROGRAM styrofoam
*     начальные условия и параметры
      CALL start(y, v, t, g, vt2, dt, height)
*     вывод параметров
      CALL prtpar(t, dt, n0, ncalc)
*     печать начальных значений
      CALL prttab(y, v, g, t)
      CALL Euler(y, v, accel, t, g, vt2, dt, n0)
*     печать результатов в момент времени t = 0
      CALL prttab(y, v, accel, t)
*     10000 выбрано условно как пример большого числа
      DO 100 i = 1,10000
*        обобщение подпрограммы из программы Fall
         CALL Euler(y, v, accel, t, g, vt2, dt, ncalc)
*        печать результатов через ncalc шагов
         CALL prttab(y, v, accel, t)
*        окончание работы программы, если y > высоты
         IF (y.gt.height) STOP
100   CONTINUE
      STOP
      END

      SUBROUTINE start(y, v, t, g, vt2, dt, height)
*     начальный момент времени (с)
      t = -0.132
*     начальное смещение (м)
      y = 0.0
*     начальная высота тела над землей
      height = 4.0
*     начальная скорость (м/с)
      v = 0.0
      WRITE(6,*) 'шаг по времени dt = '
      READ(5,*) dt
*     величина ускорения свободного падения
      g = 9.8
      WRITE(6,*) 'установившаяся скорость (м/с) = '
      READ(5,*) vterm
      vt2 = vterm*vterm
      RETURN
      END
```

```
      SUBROUTINE prtpar(t, dt, n0, ncalc)
      prtper = 0.1
*     число шагов между печатью результатов
      ncalc = prtper/dt
      n0 = -t/dt
*     заголовок
      WRITE(6,13)
13    FORMAT(5x,'время(с)',8x,'y(m)',7x,'скорость(м/с)',4x,'ускор(м/с*с)'/)
      RETURN
      END

      SUBROUTINE prttab(y, v, accel, t)
      WRITE(6,15) t, y, v, accel
15    FORMAT(4F15.5)
      RETURN
      END

      SUBROUTINE Euler(y, v, accel, t, g, vt2, dt, ncalc)
      DO 10 icalc = 1, ncalc
*        скорость в начале интервала
         y = y + v*dt
*        ось y направлена вниз
         accel = g*(1.0 - sign(1.0, v)*v*v/vt2)
         v = v + accel*dt
10    CONTINUE
      t = t + dt*ncalc
      RETURN
      END
```

ГЛАВА 4

```
      PROGRAM planet
*     движение планеты
*     для нахождения новой координаты используется новая скорость
*     описание массивов
      DIMENSION pos(2), vel(2)
      CALL start(pos, vel, GM, dt, nplot, ncalc)
*     рисуется положение "земли"
      CALL output(pos)
      DO 100 iplot = 1, nplot
         CALL Euler(pos, vel, GM, dt, ncalc)
         CALL output(pos)
100   CONTINUE
      STOP
      END

      SUBROUTINE start(pos, vel, GM, dt, nplot, ncalc)
      DIMENSION pos(2), vel(2)
*     астрономические единицы
      GM = 4.0*(3.14159)**2
      WRITE(6,*) 'шаг по времени (годы) = '
      READ(5,*) dt
      WRITE(6,*) 'полное время (годы) = '
      READ(5,*) tmax
      WRITE(6,*) 'шаг по времени между точками орбиты (годы) = '
      READ(5,*) pltper
*     число шагов по времени между точками орбиты
      ncalc = pltper/dt
      nplot = tmax/pltper
      WRITE(6,*) 'начальная координата x = '
      READ(5,*) pos(1)
*     начальные координата y и x-компонента скорости
      pos(2) = 0
      vel(1) = 0
      WRITE(6,*) 'начальная y-компонента скорости = '
      READ(5,*) vel(2)
*     допустимое максимальное значение большой полуоси
      r = 2*pos(1)
*     если экран не квадратный, то характеристическое отношение =
```

```
*       горизонтальный размер/вертикальный размер
*       значение для терминала VT100
        aspect = 1.25
*       инициализация графического пакета plot10
        CALL initt(1200)
        x = aspect*r
        CALL dwindo(-x,x,-r,r)
*       рисование солнца радиусом 0.1 в начале координат
        CALL disk(0.0,0.0,0.1)
        RETURN
        END

        SUBROUTINE disk(x0,y0,radius)
*       закрашивание диска
        angle = 0.0
        da = 3.14/200
        DO 100 i = 1,200
            x = radius*cos(angle)
            y = radius*sin(angle)
*           перемещаем курсор в очередную точку окружности
            CALL movea(x0 + x,y0 + y)
*           соединяем данную точку с противолежащей точкой окружности
            CALL drawa(x0 - x,y0 - y)
            angle = angle + da
100     CONTINUE
        RETURN
        END

        SUBROUTINE Euler(pos,vel,GM,dt,ncalc)
        DIMENSION pos(2),vel(2),accel(2)
        DO 10 icalc = 1,ncalc
            r = sqrt(pos(1)*pos(1) + pos(2) *pos(2))
            DO 5 i = 1,2
                accel(i) = -GM*pos(i)/(r*r*r)
                vel(i) = vel(i) + accel(i)*dt
                pos(i) = pos(i) + vel(i)*dt
5           CONTINUE
10      CONTINUE
        RETURN
        END
```

```
      SUBROUTINE output(pos)
*     рисование орбиты
      DIMENSION pos(2)
      CALL pointa(pos(1), pos(2))
      RETURN
      END

      PROGRAM planet2
*     солнечная система с двумя планетами
      DIMENSION pos(2,2), vel(2,2)
      CALL start(pos, vel, GM, dt, nplot, ncalc)
      CALL output(pos)
      DO 100 iplot = 1, nplot
         CALL Euler(pos, vel, GM, dt, ncalc)
         CALL output(pos)
100   CONTINUE
      STOP
      END

      SUBROUTINE start(pos, vel, GM, dt, nplot, ncalc)
      DIMENSION pos(2,2), vel(2,2)
*     астрономические единицы
      GM = 4.0*(3.14159)**2
      WRITE(6,*) 'шаг по времени (годы) = '
      READ(5,*) dt
      WRITE(6,*) 'полное время (годы) = '
      READ(5,*) tmax
      WRITE(6,*) 'промежуток времени между точками орбиты (годы) = '
      READ(5,*) pltper
*     число шагов по времени между точками орбиты
      ncalc = pltper/dt
      nplot = tmax/pltper
*     начальные координаты первой планеты
      pos(1,1) = 1.0
      pos(1,2) = 0.0
      vel(1,1) = 0.0
      vel(1,2) = sqrt(GM/pos(1,1))
```

```
*       начальные координаты второй планеты
        pos(2,1) = 4**(1/3)
        pos(2,2) = 0.0
        vel(2,1) = 0.0
        vel(2,2) = sqrt(GM/pos(2,1))
*       допустимое максимальное значение большой полуоси
        r = 2*pos(2,1)
*       если экран не квадратный, то характеристическое отношение =
*       вертикальный размер/горизонтальный размер
*       значение для терминала VT100
        aspect = 1.25
*       инициализация графического пакета plot10
        CALL initt(1200)
        x = aspect*r
        CALL dwindo(-x,x,-r,r)
*       рисование солнца радиусом 0.1 в начале координат
        CALL disk(0.0,0.0,0.1)
        RETURN
        END

        SUBROUTINE disk(x0,y0,radius)
        angle = 0.0
        da = 3.14/200
        DO 100 i = 1,200
            x = radius*cos(angle)
            y = radius*sin(angle)
*           перемещаем курсор в очередную точку окружности
            CALL movea(x0 + x,y0 + y)
*           соединяем данную точку с противолежащей точкой окружности
            CALL drawa(x0 - x,y0 - y)
            angle = angle + da
100     CONTINUE
        RETURN
        END
```

```
      SUBROUTINE Euler(pos, vel, GM, dt, ncalc)
      DIMENSION pos(2,2), vel(2,2), a(2,2), r(2)
      DO 10 icalc = 1, ncalc
*        вычисление расстояния dr между двумя планетами
         dx = pos(2,1) - pos(1,1)
         dy = pos(2,2) - pos(1,2)
         dr = sqrt(dx*dx + dy*dy)
         accel = GM/(dr*dr*dr)
*        ускорение планеты 1, обусловленное планетой 2
         a(1,1) = -0.01*accel*dx
         a(1,2) = -0.01*accel*dy
*        ускорение планеты 2, обусловленное планетой 1
         a(2,1) = -0.001*a(1,1)
         a(2,2) = -0.001*a(1,2)
         DO 5 ip = 1,2
            dist2 = pos(ip,1)**2 + pos(ip,2)**2
            r(ip) = sqrt(dist2)
            DO 2 i = 1,2
               accel = a(ip,i) - GM*pos(ip,i)/(r(ip)**3)
               vel(ip,i) = vel(ip,i) + accel*dt
               pos(ip,i) = pos(ip,i) + vel(ip,i)*dt
2           CONTINUE
5        CONTINUE
10    CONTINUE
      RETURN
      END

      SUBROUTINE output(pos)
*     рисование орбиты
      DIMENSION pos(2,2)
*     планета 1
      CALL pointa(pos(1,1), pos(1,2))
*     планета 2
      CALL pointa(pos(2,1), pos(2,2))
      RETURN
      END
```

ГЛАВА 5

```
      PROGRAM sho
*     простой гармонический осциллятор
      CALL start(pos, vel, w2, dt, ncalc)
      t = 0
*     10000 выбрано условно как пример большого числа
      DO 10 i = 1,10000
         CALL output(pos, vel, t)
         CALL Euler(pos, vel, w2, dt, ncalc)
         t = t + ncalc*dt
10    CONTINUE
      STOP
      END

      SUBROUTINE start(pos, vel, w2, dt, ncalc)
      WRITE(6,*) 'введите начальную координату (м)'
      READ(5,*) pos
*     начальная скорость (м/с)
      vel = 0.0
*     собственная (угловая) частота
      WRITE(6,*) 'введите отношение k/m'
      READ(5,*) w2
      WRITE(6,*) 'введите шаг по времени (с) ( < 0.05)'
      READ(5,*) dt
      prtper = 0.05
      ncalc = prtper/dt
      WRITE(6,15)
15    FORMAT(7x,'время',17x,'координата',28x,'скорость')
      WRITE(6,*)
      RETURN
      END
```

```
      SUBROUTINE Euler(pos, vel, w2, dt, ncalc)
*     алгоритм Эйлера—Кромера
      DO 10 icalc = 1, ncalc
         accel = -w2*pos
         vel = vel + accel*dt
         pos = pos + vel*dt
10    CONTINUE
      RETURN
      END

      SUBROUTINE output(pos, vel, t)
      WRITE(6, 25) t, pos, vel
*     каждое число занимает 20 позиций; дробная часть - 4 десят. знака
25    FORMAT(3F20.4)
      RETURN
      END

      PROGRAM rc
      CALL start(R, tau, V0, w, tmax, dt)
      CALL screen(shift, V0, tmax)
      CALL scope(shift, R, tau, V0, w, tmax, dt)
      STOP
      END

      SUBROUTINE start(R, tau, V0, w, tmax, dt)
*     амплитуда внешнего напряжения
      V0 = 1.0
      WRITE(6,*) 'частота внешнего напряжения (Гц) = '
      READ(5,*) f
*     угловая частота
      w = 2.0*3.14159*f
      WRITE(6,*) 'сопротивление (Ом) = '
      READ(5,*) R
      WRITE(6,*) 'емкость (Ф) = '
      READ(5,*) C
      WRITE(6,*) 'шаг по времени dt = '
      READ(5,*) dt
*     время релаксации
      tau = R*C
*     период напряжения внешнего источника
      T = 1/f
```

```
      IF (T.gt.tau) then
         tmax = 2*T
      ELSE
         tmax = 2*tau
      END IF
      RETURN
      END

      SUBROUTINE screen(shift, V0, tmax)
      CHARACTER*25 title
      tmin = 0.0
      Vmin = -V0
*     сдвиг экранных координат по y для верхнего графика
      shift = 2.5*V0
*     максимальное значение экранной координаты y
      ymax = shift + 1.1*(V0 - Vmin)
      CALL initt(2400)
      CALL dwindo(1.1*tmin, 1.1*tmax, 1.1*Vmin, ymax)
      title = 'источник напряжения'
      CALL axis(tmin, tmax, Vmin, V0, title)
      title = 'падение напряжения на резисторе'
      CALL axis(tmin, tmax, Vmin+shift, V0+shift)
      RETURN
      END

      SUBROUTINE scope(shift, R, tau, V0, w, tmax, dt)
      ntime = tmax/dt
      t = 0.0
*     значение при t = 0
      Vs0 = V(V0, w, t)
      Vo0 = Vs0
      Q = 0.0
      DO 100 itime = 1, ntime
         t = t + dt
         cur = V(V0, w, t)/R - Q/tau
         Q = Q + cur*dt
         Vs = V(V0, w, t)
         Vout = cur*R
         CALL movea(t-dt, Vs0)
         CALL drawa(t, Vs)
```

```
            CALL movea(t-dt,Vo0+shift)
            CALL drawa(t,Vout+shift)
            Vs0 = Vs
            Vo0 = Vout
100     CONTINUE
        RETURN
        END

        FUNCTION V(V0,w,t)
        V = V0*cos(w*t)
        RETURN
        END

        SUBROUTINE axis(xmin,xmax,ymin,ymax,title)
        CHARACTER*25 title
        ntick = 10
        dx = (xmax - xmin)/ntick
        dy = (ymax - ymin)/ntick
*       определение положения осей
        IF (xmin*xmax.lt.0.0) then
            x0 = 0.0
        ELSE
            x0 = xmin
        END IF
        IF (ymin*ymax.lt.0.0) then
            y0 = 0.0
        ELSE
            y0 = ymin
        END IF
*       построение вертикальной и горизонтальной осей
        CALL movea(x0,ymin)
        CALL drawa(x0,ymax)
        CALL movea(xmin,y0)
        CALL drawa(xmax,y0)
*       размер вертикального деления на оси x
        xlen = 0.1*dy
*       размер горизонтального деления на оси y
        ylen = 0.1*dx
*       нанесение делений на оси
```

```
      DO 50 i = 0,ntick-1
         col = xmin + i*dx
         row = ymin + i*dy
         CALL movea(col,y0-xlen)
         CALL drawa(col,y0+xlen)
         CALL movea(x0-ylen,row)
         CALL drawa(x0+ylen,row)
50    CONTINUE
*     печать заголовка
      CALL movea(xmin + 7.0*dx,ymax)
      WRITE(6,11) title
11    FORMAT(a25)
*     вывод максимальных значений x и y
      CALL movea(xmax-0.5*xlen,y0)
      WRITE(6,13) xmax
      CALL movea(x-4.0*ylen,ymax+ylen)
      WRITE(6,13) ymax
13    FORMAT(f5.1)
      RETURN
      END
```

ГЛАВА 6

```
      PROGRAM md
      DIMENSION x(1024),y(1024),vx(1024),vy(1024)
      DIMENSION ax(1024),ay(1024)
      CALL start(x,y,vx,vy,N,Sx,Sy,dt,dt2,nsnap,ntime)
      CALL accel(x,y,ax,ay,N,Sx,Sy,virial,zpe)
      virial = 0.0
      zpe = 0.0
      DO 100 isnap = 1,nsnap
        DO 10 itime = 1,ntime
          CALL move(x,y,vx,vy,ax,ay,N,Sx,Sy,dt,dt2,flx,fly,virial,zke,zpe)
10      CONTINUE
        CALL output(flx,fly,virial,zke,zpe,Sx,Sy,dt,N,ntime)
100   CONTINUE
      STOP
      END
```

```
      SUBROUTINE start(x, y, vx, vy, N, Sx, Sy, dt, dt2, nsnap, ntime)
      DIMENSION x(1024), y(1024), vx(1024), vy(1024)
      WRITE(6,*) 'введите число частиц'
      READ(5,*) N
      WRITE(6,*) 'введите размеры ящика'
      READ(5,*) Sx, Sy
      WRITE(6,*) 'введите шаг по времени'
      READ(5,*) dt
      dt2 = dt*dt
      WRITE(6,*) 'введите максимальное значение скорости'
      READ(5,*) vmax
      WRITE(6,*) 'введите число снимков и число шагов между ними'
      READ(5,*) nsnap, ntime
      WRITE(6,*) 'введите отрицательное случайное начальное число'
      READ(5,*) iseed
      WRITE(6,*) 'введите конфигурацию: 1=старая, 2=горячая, 3=холодная'
      READ(5,*) icf
      IF (icf.eq.1) then
*        чтение старой конфигурации
         DO 10 i = 1,N
            READ(8,12) x(i), y(i), vx(i), vy(i)
12          FORMAT(4(2x, f10.5))
10       CONTINUE
      ELSEIF (icf.eq.3) then
*        упорядоченная (холодная) начальная конфигурация
         area1 = Sx*Sy/N
         ys = 0.5*sqrt(3.0)
         a = sqrt(area1/ys)
         Ly = 2*int(.5*(1.0 + Sy/(a*ys)))
         Lx = N/Ly
         DO 30 ix = 1,Lx
            DO 20 iy = 1,Ly
               i = (iy - 1)*Ly + ix
               y(i) = (iy - 0.5)*a*ys
               IF (mod(iy,2).eq.0) then
                  x(i) = (ix - 0.25)*a
               ELSE
                  x(i) = (ix - 0.75)*a
               END IF
```

```
                vx(i) = vmax*(2*ran(iseed) - 1)
                vy(i) = vmax*(2*ran(iseed) - 1)
20          CONTINUE
30        CONTINUE
        ELSE
*         случайная (горячая) конфигурация
          DO 40 i = 1,N
            x(i) = Sx*ran(iseed)
            y(i) = Sy*ran(iseed)
            vx(i) = vmax*(2*ran(iseed) - 1)
            vy(i) = vmax*(2*ran(iseed) - 1)
40        CONTINUE
        ENDIF
        DO 50 i = 1,N
          vxcum = vxcum + vx(i)
          vycum = vycum + vy(i)
50      CONTINUE
        vxcum = vxcum/N
        vycum = vycum/N
        DO 60 i = 1,N
          vx(i) = vx(i) - vxcum
          vy(i) = vy(i) - vycum
60      CONTINUE
        RETURN
        END

        SUBROUTINE move(x,y,vx,vy,ax,ay,N,Sx,Sy,dt,dt2,flx,fly,virial,zke,zpe)
        DIMENSION x(1024),y(1024),vx(1024),vy(1024)
        DIMENSION ax(1024),ay(1024)
        DO 10 i = 1,N
          xnew = x(i) + vx(i)*dt + 0.5*ax(i)*dt2
          ynew = y(i) + vy(i)*dt + 0.5*ay(i)*dt2
          CALL cellp(xnew,ynew,vx(i),vy(i),Sx,Sy,flx,fly)
          x(i) = xnew
          y(i) = ynew
          vx(i) = vx(i) + 0.5*ax(i)*dt
          vy(i) = vy(i) + 0.5*ay(i)*dt
10      CONTINUE
        CALL accel(x,y,ax,ay,N,Sx,Sy,virial,zpe)
```

```
      DO 20 i = 1,n
         vx(i) = vx(i) + 0.5*dt*ax(i)
         vy(i) = vy(i) + 0.5*dt*ay(i)
         zke = zke + vx(i)*vx(i) + vy(i)*vy(i)
         virial = virial + ax(i)*x(i) + ay(i)*y(i)
20    CONTINUE
      RETURN
      END

      SUBROUTINE accel(x,y,ax,ay,N,Sx,Sy,virial,zpe)
      DIMENSION x(1024),y(1024),ax(1024),ay(1024)
      DO 1 i = 1,n
         ax(i) = 0.0
         ay(i) = 0.0
1     CONTINUE
      DO 20 i = 1,(n-1)
         DO 10 j = (i+1),n
            dx = x(i) - x(j)
            dy = y(i) - y(j)
            CALL sep(dx,dy,Sx,Sy)
            r = sqrt(dx*dx + dy*dy)
            CALL FandU(r,force,pot)
            ax(i) = ax(i) + force*dx
            ay(i) = ay(i) + force*dy
            virial = virial + force*r*r
            zpe = zpe + pot
*     определение силы, действующей на частицу j по 3-му закону Ньютона
            ax(j) = ax(j) - force*dx
            ay(j) = ay(j) - force*dy
10       CONTINUE
20    CONTINUE
      RETURN
      END
```

```
SUBROUTINE FandU(r, force, pot)
ri = 1/r
ri3 = ri*ri*ri
ri6 = ri3*ri3
ri12 = ri6*ri6
g = 24*ri*ri6*(2*ri6 - 1)
force = g*ri
pot = 4*(ri12 - ri6)
RETURN
END

SUBROUTINE sep(dx, dy, Sx, Sy)
IF (abs(dx).gt.0.5*Sx) dx = dx - sign(Sx, dx)
IF (abs(dy).gt.0.5*Sy) dy = dy - sign(Sy, dy)
RETURN
END

SUBROUTINE cell(xnew, ynew, Sx, Sy)
IF (xnew.lt.0) xnew = xnew + Sx
IF (xnew.gt.Sx) xnew = xnew - Sx
IF (ynew.lt.0) ynew = ynew + Sy
IF (ynew.gt.Sy) ynew = ynew - Sy
RETURN
END

SUBROUTINE cellp(xnew, ynew, vx, vy, Sx, Sy, flx, fly)
IF (xnew.lt.0) then
    xnew = xnew + Sx
    flx = flx - vx
END IF
IF (xnew.gt.Sx) then
    xnew = xnew - Sx
    flx = flx + vx
END IF
IF (ynew.lt.0) then
    ynew = ynew + Sy
    fly = fly - vy
END IF
```

```
      IF (ynew.gt.Sy) then
         ynew = ynew - Sy
         fly = fly + vy
      END IF
      RETURN
      END

      SUBROUTINE output(flx, fly, virial, zke, zpe, Sx, Sy, dt, N, ntime)
      data iff /0/
      IF (iff.eq.0) then
         iff = 1
         write(6,*) 'ke   pe   tot    pflux   pvirial    pideal'
      END IF
      pflux = ((flx/Sx) + (fly/Sy))/(2*dt*ntime)
      zke = 0.5*zke/ntime
      zpe = zpe/ntime
      tot = zke + zpe
      pideal = zke/(Sx*Sy)
      pvirial = pideal + (.5/(Sx*Sy))*virial/ntime
      write(6,13) zke, zpe, tot, pflux, pvirial, pideal
13    format(6(1x,e13.6))
      zke = 0
      zpe = 0
      fly = 0
      flx = 0
      virial = 0
      RETURN
      END
```

ГЛАВА 7

```
      PROGRAM maptable
      character*1 choice
*     итерации одномерного отображения f(x)
*     idum — большое число
```

```
      DO 100 idum = 1,10000
         CALL start(x,r)
         CALL map(x,r)
         WRITE(6,*) 'продолжать? (y/n)'
         READ(5,13) choice
13       FORMAT(a1)
         IF (choice.eq.'n') STOP
100   CONTINUE
      STOP
      END

      SUBROUTINE start(x,r)
      WRITE(6,*) 'параметр роста (0 < r < 1) = '
      READ(5,*) r
      WRITE(6,*) 'начальное значение x (0 < x < 1) = '
      READ(5,*) x
      RETURN
      END

      SUBROUTINE map(x,r)
      DIMENSION xs(8)
      CHARACTER*1 CHOICE
      iter = 0
      DO 100 idum = 1,1000
*        печать 20 строк по 8 итераций в каждой
         DO 50 irow = 1,20
            DO 10 i = 1,8
               xs(i) = 4*r*x*(1 - x)
               x = xs(i)
               iter = iter + 1
10          CONTINUE
            WRITE(6,15) (xs(k),k=1,8)
15          FORMAT(8f9.6)
50       CONTINUE
         WRITE(6,*) 'число итераций = ', iter
         WRITE(6,*) 'еще итерировать? (y/n)'
         READ(5,17) choice
17       FORMAT(a1)
```

```
            IF (choice.ne.'y') RETURN
100     CONTINUE
        RETURN
        END
```

ГЛАВА 8

```
        PROGRAM oscillators
*       вычисляется смещение N связанных осцилляторов
        DIMENSION u(0:21), vel(20)
        CALL start(N, u, vel, kc, k, dt, tmax)
        CALL screen(N, nplot, tmax, dy)
        CALL move(N, u, vel, kc, k, dt, nplot, tmax, dy)
        STOP
        END

        SUBROUTINE start(N, u, vel, kc, k, dt, tmax)
        DIMENSION u(0:21), vel(20)
        WRITE(6,*) 'число частиц = '
        READ(5,*) N
        WRITE(6,*) 'шаг по времени = '
        READ(5,*) dt
        WRITE(6,*) 'продолжительность = '
        READ(5,*) tmax
        WRITE(6,*) 'силовая постоянная kc = '
        READ(5,*) kc
*       прилипают к стенкам ?
        k = 1
*       начальные смещения первых двух осцилляторов
        u(1) = 0.5
        u(2) = 0.0
*       начальные скорости полагаются равными нулю
        DO 10 i = 1,N
            vel(i) = 0
10      CONTINUE
        RETURN
        END
```

```
      SUBROUTINE screen(N, nplot, tmax, dy)
*     расстояние между графиками на экране
      dy = 2
      CALL initt(1200)
      CALL dwindo(-1.0, tmax + 1.0, -dy, 3.0*dy)
      ntick = 100
*     расстояние между делениями
      dx = tmax/ntick
*     "высота" делений
      hy = 0.1*dx
*     число осцилляторов для рисования графиков
      nplot = min(4, N)
*     рисуем деления оси
      sy = -1.0
      DO 30 j = 1, nplot
         CALL movea(0.0, sy)
         CALL drawa(tmax, sy)
         DO 20 i = 1, ntick
            sx= i*dx
            CALL movea(sx, sy)
            CALL drawa(sx, sy+hy)
20       CONTINUE
         sy = sy + dy
30    CONTINUE
      RETURN
      END

      SUBROUTINE move(N, u, vel, kc, k, dt, nplot, tmax, dy)
      DIMENSION u(0:21), vel(20)
      DIMENSION a(20)
      DO 1000 idum = 1, 1000
         t = t + dt
*        ускорение точек, не связанных со стенками
         DO 40 i = 2, (N - 1)
            a(i) = kc*(u(i + 1) + u(i - 1) - 2*u(i))
40       CONTINUE
*        ускорения концевых масс
         a(1) = kc*(u(2) - u(1)) - k*u(1)
         a(N) = kc*(u(N - 1) - u(N)) - k*u(N)
```

```
*           алгоритм Эйлера — Кромера
            DO 50 i = 1,N
               vel(i) = vel(i) + a(i)*dt
               u(i) = u(i) + vel(i)*dt
50          CONTINUE
            row = -1
            DO 60 i = 1,nplot
               CALL pointa(t,u(i)+row)
               row = row + dy
60          CONTINUE
            IF (t.ge.tmax) RETURN
1000    CONTINUE
        RETURN
        END
```

ГЛАВА 9

```
        PROGRAM Laplace
        DIMENSION V(100,100)
        CALL assign(V,nx,ny,change)
        CALL relax(V,nx,ny,change,iter)
        STOP
        END

        SUBROUTINE assign(V,nx,ny,change)
        DIMENSION V(100,100)
        WRITE(6,*) 'число ячеек по оси x = '
        READ(5,*) nx
        WRITE(6,*) 'число ячеек по оси y = '
        READ(5,*) ny
        WRITE(6,*) 'потенциал на границе прямоугольника в вольтах = '
        READ(5,*) V0
        WRITE(6,*) 'минимальная относительная точность сходимости = '
        READ(5,*) change
        change = change/100
        DO 100 i = 1,nx
*           фиксируем потенциал на границе
            V(i,1) = V0
            V(i,ny) = V0
100     CONTINUE
```

```
      DO 200 j = 1,nγ
         V(1,j) = V0
         V(nx,j) = V0
200   CONTINUE
*     задаем начальные потенциалы внутренних ячеек
      DO 400 i = 2,nx - 1
         DO 300 j = 2,nγ - 1
            V(i,j) = 0.9*V0
300      CONTINUE
400   CONTINUE
      RETURN
      END

      SUBROUTINE relax(V,nx,nγ,change,iter)
      DIMENSION Vave(100,100),V(100,100)
      iter = 0
      DO 1000 idum = 1,1000
         dmax = 0
         iter = iter + 1
         DO 200 i = 2,nx - 1
            DO 100 j = 2,nγ - 1
*              вычисляем средний потенциал соседних ячеек
               Vave(i,j) = V(i+1,j) + V(i-1,j)
               Vave(i,j) = Vave(i,j) + V(i,j+1) + V(i,j-1)
               Vave(i,j) = 0.25*Vave(i,j)
*              вычисляем относительное изменение потенциала
               diff = abs((V(i,j) - Vave(i,j))/Vave(i,j))
               IF (diff.gt.dmax) dmax = diff
100         CONTINUE
200      CONTINUE
*        изменяем потенциал в каждой ячейке
         DO 400 i = 2,nx - 1
            DO 300 j = 2,nγ - 1
               V(i,j) = Vave(i,j)
300         CONTINUE
400      CONTINUE
         CALL output(V,nx,nγ,iter)
```

```
*           повторяем до требуемой точности
            IF (dmax.lt.change) RETURN
1000    CONTINUE
        RETURN
        END

        SUBROUTINE output(V,nx,ny,iter)
        DIMENSION V(100,100)
*       печать результатов
        WRITE(6,*) 'число итераций = ', iter
        DO 100 j = ny,1,-1
            WRITE(6,13) (V(i,j), i = 1,nx)
13          FORMAT(10F8.4)
100     CONTINUE
        RETURN
        END
```

ПРИЛОЖЕНИЕ Д. РАСПЕЧАТКИ ПРОГРАММ НА ЯЗЫКЕ ПАСКАЛЬ: ЧАСТЬ 1

Ниже приводятся версии перевода большинства программ из ч. 1 данной книги с языка True BASIC на Паскаль. Программы пропускались с использованием компилятора Macintosh Pascal. Для других версий Паскаля необходимо внести минимальные изменения.

Поскольку язык Паскаль разрабатывался в те годы, когда машинная графика еще не получила широкого распространения, то в него не вошли графические процедуры. В языке True BASIC программист может устанавливать экранную систему координат по правилу, принятому в математике, согласно которому наибольшее значение вертикальной координаты отвечает верхней границе экрана. В противоположность этому в графических процедурах, имеющихся в большинстве версий Паскаля для микрокомпьютеров, принято другое соглашение, и в них требуется, чтобы верхней границе экрана отвечало наименьшее значение вертикальной координаты. Хотя можно написать процедуры перевода графических операций, описанных в микрокомпьютерных версиях Паскаля, в операции, аналогичные применяемым в языке True BASIC, мы связали координаты x и y с соответствующими номерами пикселов экрана для машины Macintosh.

Применяемые в программах на Паскале графические процедуры входят в состав стандартного программного обеспечения компьютера Macintosh и могут вызываться из других языков программирования, реализованных на этом компьютере. Краткое описание графических процедур, используемых в наших программах, приведено в табл. Д1.

ТАБЛИЦА Д1. Список стандартных графических процедур языка Macintosh Pascal

Процедура	Описание
moveto(x,y)	Помещает перо в точку (x,y)
lineto(x2,y2)	Проводит прямую из текущей точки в точку с координатами (x2,y2)
paintoval(top,left,bottom,right)	Строит закрашенный эллипс, вписанный в прямоугольник с целочисленными координатами top, left, bottom, right
framerect(top,left,bottom,right)	Рисует прямоугольник
eraserect(top,left,bottom,right)	Стирает заполнение прямоугольника
paintrect(top,left,bottom,right)	Закрашивает прямоугольник
drawstring(strg)	Выводит строку, начиная с текущей позиции

В табл. Д2 приводятся некоторые важные графические процедуры, применяемые в системе Turbo Pascal (Borland International), реализованной на компьютерах IBM PC и совместимых с ними.

ТАБЛИЦА Д2. Список «основных» графических процедур системы Turbo Pascal для IBM PC

Процедура	Описание
HiRes	Режим высокого разрешения (640 × 200 точек)
GraphMode	Монохромная графика (320 × 200 точек)
GraphColorMode	Цветная графика (320 × 200 точек)
TextMode	25 строк по 80 символов в каждой
Plot(x,y,color)	Рисует точку с координатами (x,y)
Draw(x1,y1,x2,y2,color)	Проводит прямую из точки (x1,y1) в точку (x2,y2)
Circle(x,y,color)	Проводит окружность заданного радиуса с центром в точке (x,y)
FillShape(x,y,fc,bc)	Закрашивает область заданным цветом fc; точка (x,y) лежит внутри замкнутого контура, цвет которого задается параметром bc

ГЛАВА 2

```
program example(input,output);
var                                    (* описания переменных *)
   y, x, dx : real;
   n : integer;

   procedure initial (var y, x, dx : real;
                      var n : integer);
   var
      xmax : real;
   begin
     x := 1.0;                         (* начальное значение x *)
     xmax := 2.0;                      (* максимальное значение x *)
     y := 1;                           (* начальное значение y *)
     dx := 0.1;                        (* величина шага *)
     n := trunc((xmax - x) / dx)
   end;
```

```
    procedure Euler (var y, x, dx : real;
                     var n : integer);
    var
       slope, change : real;
       i : integer;
    begin
      for i := 1 to n do                (* цикл повторяется n раз *)
          begin
            slope := 2.0 * x;           (* наклон в начальной точке *)
            change := slope * dx;       (* оценка изменения функции *)
            y := y + change;            (* новое значение y *)
            x := x + dx;                (* приращение величины x *)
            writeln(x, y)
          end
    end;

begin                          (* основная программа *)
  initial(y, x, dx, n);
  Euler(y, x, dx, n)
end.

program cool;       (* метод Эйлера для задачи об остывании кофе *)
var                 (* описания переменных *)
    t, temperature, room_temp, r, dt : real;
    ncalc : integer;

    procedure initial (var t, temperature, room_temp, r, dt : real;
                       var ncalc : integer);
    var
       tmax : real;
    begin
      t := 0.0;                         (* начальный момент времени *)
      temperature := 83;                (* начальная температура кофе (C) *)
      room_temp := 22;                  (* комнатная температура (C) *)
      r := 0.1;                         (* коэффициент остывания (1/мин) *)
      dt := 0.1;                        (* шаг по времени (мин) *)
      tmax := 2.0;                      (* длительность наблюдения (мин) *)
      ncalc := trunc(tmax / dt)         (* общее количество шагов *)
    end;
```

```
    procedure output (t, temperature : real);
    begin
      writeln(t, temperature)          (* печать результатов *)
    end;

    procedure Euler (var t, temperature, room_temp, r, dt : real;
                     var ncalc : integer);
    var
       change : real;
       icalc : integer;
    begin
      for icalc := 1 to ncalc do       (* цикл повторяется ncalc раз *)
         begin
           change := -r * (temperature - room_temp);
           temperature := temperature + change * dt;
           t := t + dt;                (* время *)
           output(t, temperature)
         end
    end;

begin                          (* основная программа *)
  initial(t, temperature, room_temp, r, dt, ncalc);
  Euler(t, temperature, room_temp, r, dt, ncalc)
end.
```

ГЛАВА 3

```
program fall (input, output);          (* свободно падающее тело *)
const
  g = 9.8;                  (* величина ускорения свободного падения *)
var
    y, v, accel, t, dt, height : real;
    ncalc : integer;
```

```
procedure initial (var y, v, t, dt, height : real);
begin
  t := 0.0;              (* начальное время (с) *)
  y := 0.0;              (* начальное смещение (м) *)
  height := 10.0;        (* начальная высота тела над землей *)
  v := 0.0;              (* начальная скорость *)
  write('шаг по времени dt = ');
  readln(dt)
end;

procedure print_parameters (var dt : real;
                           var ncalc : integer);
var
    print_period : real;
begin
  print_period := 0.1;                  (* (с) *)
  ncalc := round(print_period / dt); (* число шагов между печатями *)
  write('время (с)       ');           (* заголовок *)
  write('y (м)     ');
  write('скорость (м/с)     ');
  writeln('ускорение (м/с*с)');
  writeln
end;

procedure print_table (y, v, accel, t : real);
(* печать результатов в табличном виде *)
begin
  writeln(t : 6 : 2, y : 12 : 2, v : 12 : 2, accel : 12 : 2)
end;
```

```
    procedure Euler (var y, v, accel, t, dt : real;
                     ncalc : integer);
    var
      icalc : integer;
    begin
      for icalc := 1 to ncalc do
          begin
            y := y + v * dt;
            accel := g;
            v := v + accel * dt;
          end;
      t := t + dt * ncalc;
    end;

begin                          (* основная программа *)
  initial(y, v, t, dt, height);          (* начальные условия и параметры *)
  print_parameters(dt, ncalc);
  print_table(y, v, g, t);               (* печать начальных условий *)
  repeat
      Euler(y, v, accel, t, dt, ncalc);
      print_table(y, v, accel, t);
  until y > height;
end.
```

ГЛАВА 4

```
program planet (input, output);
(* движение планеты *)
(* для нахождения новой координаты используется новая скорость *)
type vector = array[1..2]  of  real;    (* описание размера массивов *)
var
    pos, vel : vector;
    GM, dt, rmax : real;
    ncalc, nplot, iplot : integer;
```

```
procedure initial (var pos, vel : vector;
                   var GM, dt, rmax : real;
                   var nplot, ncalc : integer);
const
    pi = 3.14159;
var
    plot_period, tmax : real;
begin
   GM := 4.0 * pi * pi;          (* астрономические единицы *)
   writeln('шаг по времени = ');
   readln(dt);
   writeln('длительность наблюдения (годы) = ');
   readln(tmax);
   writeln('период графика (годы) = ');
   readln(plot_period);
   (* число шагов по времени между точками орбиты *)
   ncalc := round(plot_period / dt);
   nplot := round(tmax / plot_period);
   writeln('начальная координата x = ');
   readln(pos[1]);
   rmax := 2.0 * pos[1];          (* максимум большой полуоси *)
   pos[2] := 0.0;                 (* начальное значение координаты y *)
   vel[1] := 0.0;                 (* начальное значение скорости x *)
   writeln('начальная y-компонента скорости = ');
   readln(vel[2]);
   paintoval(130,230,170,270)     (* рисуем солнце в начале координат *)
end;
```

```
procedure Euler (var pos, vel : vector;
                 GM, dt : real;
                 ncalc : integer);
var
   accel : vector;
   icalc, i : integer;
   r : real;
begin
  for icalc := 1 to ncalc do
    begin
      r := sqrt(pos[1] * pos[1] + pos[2] * pos[2]);
      writeln(r);
      for i := 1 to 2 do
          begin
            accel[i] := -GM * pos[i] / (r * r * r);
            vel[i] := vel[i] + accel[i] * dt;
            pos[i] := pos[i] + vel[i] * dt
          end
    end
end;

procedure orbit (pos : vector;
                 rmax : real);
(* рисование орбиты *)
const
   x0 = 250;
   y0 = 150;
   aspect = 0.6667
var
   i, j : integer;
begin
  i := round(x0 + (250 / rmax) * pos[1] * aspect);
  j := round(y0 - (150 / rmax) * pos[2]);
  paintoval(j - 1,i - 1,j + 1,i + 1)
end;
```

```
begin                          (* основная программа *)
  initial(pos, vel, GM, dt, rmax, nplot, ncalc);
  orbit(pos, rmax);
  for iplot := 1 to nplot do
      begin
        Euler(pos, vel, GM, dt, ncalc);
        orbit(pos, rmax)
      end
end.
```

ГЛАВА 5

```
program sho (input, output);  (* простой гармонический осциллятор *)
var
    dataout : text;
    pos, vel, w2, dt, t : real;
    ncalc, iprt, nprt : integer;

    procedure initial (var pos, vel, w2, dt : real;
                       var ncalc, nprt : integer);
    var
        prt_period, total_time : real;
        fname : string[10];
    begin
      write('начальная координата (метры) = ');
      readln(pos);
      vel := 0.0;                    (* начальная скорость (м/с) *)
      write('отношение k/m = ');
      readln(w2);
      write('шаг по времени (с) = ');
      readln(dt);
      write('время между печатью данных = ');
      readln(prt_period);
      write('полное время моделирования = ');
      readln(total_time);
      ncalc := round(prt_period / dt);
      nprt := round(total_time / prt_period);
      write('имя файла данных = ');
      readln(fname);
```

```
    open(dataout, fname);
    rewrite(dataout);
    writeln(dataout, 'время' : 11, 'координата' : 11, 'скорость' : 11);
    writeln(dataout);
    writeln('время' : 11, 'координата' : 11, 'скорость' : 11);
    writeln
  end;

  procedure Euler (var pos, vel, w2, dt : real;
                   ncalc : integer);  (* алгоритм Эйлера-Кромера *)
  var
    accel : real;
    icalc : integer;
  begin
    for icalc := 1 to ncalc do
      begin
        accel := -w2 * pos;
        vel := vel + accel * dt;
        pos := pos + vel * dt
      end
  end;

  procedure outdat (pos, vel, t : real);
  begin
    writeln(t : 11 : 4, pos : 11 : 4, vel : 11 : 4);
    writeln(dataout, t : 11 : 4, pos : 11 : 4, vel : 11 : 4)
  end;

begin                          (* основная программа *)
  initial(pos, vel, w2, dt, ncalc, nprt);
  t := 0.0;
  for iprt := 1 to nprt do
    begin
      outdat(pos, vel, t);
      Euler(pos, vel, w2, dt, ncalc);
      t := t + ncalc * dt
    end;
  close(dataout)
end.
```

ГЛАВА 6

```
program md (input, output);
const
    c = 32768              (* нужно для получения случайных чисел *)
    Nmax = 30;             (* максимальное число частиц *)
type
    component = array[1..Nmax] of real;
var
    x, y, vx, vy, ax, ay : component;
    N, nave, nset, iset, iave : integer;
    Lx, Ly, dt, dt2 : real;
    virial, xflux, yflux, pe, ke, time : real;

    procedure initial (var x, y, vx, vy : component;
                       var N, nave, nset : integer;
                       var Lx, Ly, dt, dt2 : real);
    var
        irow, icol, nrow, ncol, i : integer;
        ax, ay, xscale, yscale, Mx, My : real;
        vscale, vmax, vxcum, vycum : real;
        fname : nstring;
        datain : text;
        newconf : char;
    begin
      write('число частиц = ');
      readln(N);
      write('размеры ящика Lx и Ly = ');
      readln(Lx, Ly);
      write('шаг по времени = ');
      readln(dt);
      dt2 := dt * dt;
      write('число шагов по времени между усреднениями = ');
      readln(nave);
      write('количество наборов усреднения = ');
      readln(nset);
      write('новая конфигурация? (y/n)');
      readln(newconf);
```

```
if (newconf = 'y') then
    begin                    (* начинаем на треугольной решетке *)
      write('число частиц в ряду = ');
      readln(nrow);
      write('максимальная скорость = ');
      readln(vmax);
      ncol := N div nrow;
      (* расстояние до ближайшего соседа в каждом направлении *)
      ay := Ly / nrow;
      ax := Lx / ncol;
      i := 0;
      for icol := 1 to ncol do
        for irow := 1 to nrow do
          begin
            i := i + 1;
            y[i] := ay * (irow - 0.5);
            if ((irow mod 2) = 0) then
                x[i] := ax * (icol - 0.25)
            else
                x[i] := ax * (icol - 0.75);
            vx[i] := random * vmax;   (* случайные скорости *)
            vy[i] := random * vmax;
            (* суммируем vx и vy, чтобы полная скорость была 0 *)
            vxcum := vxcum + vx[i];
            vycum := vycum + vy[i]
          end;
      vxcum := vxcum / N;
      vycum := vycum / N;
      for i := 1 to N do
        begin          (* полную скорость делаем равной 0 *)
          vx[i] := vx[i] - vxcum;
          vy[i] := vy[i] - vycum
        end;
    end
```

```
    else                                (* старая конфигурация *)
        begin
          write('имя файла с конфигурацией ');
          readln(fname);
          write('относительное изменение скорости = ');
          readln(vscale);
          open(datain, fname);
          reset(datain);
          readln(datain, N, Mx, My);
          xscale := Lx / Mx;
          yscale := Ly / My;
          for i := 1 to N do
              begin
                readln(datain, x[i], y[i], vx[i], vy[i]);
                x[i] := x[i] * xscale;
                y[i] := y[i] * yscale;
                vx[i] := vx[i] * vscale;
                vy[i] := vy[i] * vscale
              end;
          close(datain)
        end
end;
```

```
procedure periodic (var xtemp, ytemp, xflux, yflux : real;
                    px, py, Lx, Ly : real);
begin
  if xtemp < 0.0 then
     begin
        xtemp := xtemp + Lx;
        xflux := xflux - px
     end;
  if xtemp > Lx then
     begin
        xtemp := xtemp - Lx;
        xflux := xflux + px
     end;
  if ytemp < 0.0 then
     begin
        ytemp := ytemp + Ly;
        yflux := yflux - py
     end;
  if ytemp > Ly then
     begin
        ytemp := ytemp - Ly;
        yflux := yflux + py
     end
end;

procedure separation (var dx, dy : real;
                      Lx, Ly : real);
begin
   if abs(dx) > 0.5 * Lx then
      dx := dx * (1.0 - Lx / abs(dx));
   if abs(dy) > 0.5 * Ly then
      dy := dy * (1.0 - Ly / abs(dy))
end;

procedure f (r : real;
             var force, potential : real);
var
   ri, ri3, ri6, g : real;
```

```
begin
    ri := 1.0 / r;
    ri3 := ri * ri * ri;
    ri6 := ri3 * ri3;
    g := 24.0 * ri * ri6 * (2.0 * ri6 - 1.0);
    force := g * ri;
    potential := 4.0 * ri6 * (ri6 - 1.0)
end;

procedure accel (var x, y, ax, ay : component;
                 N : integer;
                 Lx, Ly : real;
                 var pe : real);
var
    i, j : integer;
    dx, dy, r, force, potential : real;
begin
  for i := 1 to N do
      begin
          ax[i] := 0.0;
          ay[i] := 0.0
      end;
  for i := 1 to (N - 1) do
      for j := (i + 1) to N do
          begin
              dx := x[i] - x[j];
              dy := y[i] - y[j];
              separation(dx, dy, Lx, Ly);
              r := sqrt(dx * dx + dy * dy);
              f(r, force, potential);
              ax[i] := ax[i] + force * dx;
              ay[i] := ay[i] + force * dy;
              ax[j] := ax[j] - force * dx;
              ay[j] := ay[j] - force * dy;
              pe := pe + potential
          end
end;
```

```
procedure Verlet (var x, y, vx, vy, ax, ay : component;
                  N : integer;
                  Lx, Ly, dt, dt2 : real;
                  var virial, xflux, yflux, pe, ke : real);
var
   i : integer;
   xnew, ynew : real;
begin
  for i := 1 to N do
     begin
       xnew := x[i] + vx[i] * dt + 0.5 * ax[i] * dt2;
       ynew := y[i] + vy[i] * dt + 0.5 * ay[i] * dt2;
       (* частично меняем скорость, используя старое ускорение *)
       vx[i] := vx[i] + 0.5 * ax[i] * dt;
       vy[i] := vy[i] + 0.5 * ay[i] * dt;
       (* периодические краевые условия и вычисление потока *)
       periodic(xnew, ynew, xflux, yflux, vx[i], vy[i], Lx, Ly);
       x[i] := xnew;
       y[i] := ynew
     end;
  accel(x, y, ax, ay, N, Lx, Ly, pe);    (* расчет нового ускорения *)
  for i := 1 to N do
     (* окончательно меняем скорость, используя новое ускорение *)
     begin
       vx[i] := vx[i] + 0.5 * ax[i] * dt;
       vy[i] := vy[i] + 0.5 * ay[i] * dt;
       ke := ke + 0.5 * (vx[i] * vx[i] + vy[i] * vy[i]);
       virial := virial + x[i] * ax[i] + y[i] * ay[i]
     end
end;

procedure results (N, nave : integer;
                   Lx, Ly, dt : real;
                   var virial, ke, pe, xflux, yflux, time : real);
var
   pflux, pvirial, E, T : real;
begin
  if time = 0.0 then
     writeln('время,T,E,pflux,pvirial');
     time := time + dt * nave;
```

```
        ke := ke / nave;
        pe := pe / nave;
        E := (pe + ke) / N;   (* энергия, приходящаяся на частицу *)
        T := ke / N;          (* температура *)
        ke := 0.0;
        pe := 0.0;
        (* приведенное давление из вычисления потока *)
        pflux := ((xflux / (2.0 * Lx)) + (yflux / (2.0 * Ly))) / (dt * nave);
        xflux := 0.0;
        yflux := 0.0;
        (* приведенное давление из вычисления вириала *)
        pvirial := (N * T) / (Lx * Ly) + 0.5 * virial / (Nave * Lx * Ly);
        virial := 0.0;
        writeln(time : 9 : 3, T : 9 : 4, E : 9 : 4, pflux : 9 : 4, pvirial : 9: 4)
    end;

    procedure save_conf (var x, y, vx, vy : component;
                         N : integer;
                         Lx, Ly : real);
    var
        fname : nstring;
        dataout : text;
        i : integer;
    begin
      write('имя файла с конфигурацией ');
      readln(fname);
      open(dataout, fname);
      rewrite(dataout);
      writeln(dataout, N, Lx, Ly);
      for i := 1 to N do
      writeln(dataout, x[i], y[i], vx[i], vy[i]);
      close(dataout)
      end;

begin                          (* основная программа *)
  initial(x, y, vx, vy, N, nave, nset, Lx, Ly, dt, dt2);
  pe := 0.0;
  accel(x, y, ax, ay, N, Lx, Ly, pe);
  time := 0.0;
  pe := 0.0;
```

```
    ke := 0.0;
    xflux := 0.0;
    yflux := 0.0;
    virial := 0.0;
    for iset := 1 to nset do
        begin
          for iave := 1 to nave do
            Verlet(x, y, vx, vy, ax, ay, N, Lx, Ly, dt, dt2, virial, xflux, yflux, pe, ke);
            results(N, nave, Lx, Ly, dt, virial, ke, pe, xflux, yflux, time)
        end;
    save_conf(x, y, vx, vy, N, Lx, Ly)
end.
```

ГЛАВА 7

```
program map_graph (input, output);
var
    x, r : real;
    iterate : integer;

    procedure parameter (var x, r : real;
                         var interate : integer);
    begin
      write('r = ');
      readln(r);
      write('начальное значение x = ');
      readln(x);
      write('итерации  f(x) = ');
      readln(iterate)
    end;
```

```
function f (x, r : real;
            iterate : integer) : real;
(* f определяется рекурсивно *)
var
   y : real;
begin
  if iterate > 1 then
    begin
      y := f(x, r, iterate - 1);
      f := 4 * r * y * (1.0 - y)
    end
  else
    f := 4 * r * x * (1 - x)
end;

procedure map (r : real;
               iterate : integer);
const
   min = 50;
   max = 250;
   scale = 200;
var
   delta, margin, x, y : real;
   m, n, i, j, i0, j0 : integer;
begin
  n := 200;       (* количество точек, в которых вычисляется f(x) *)
  delta := 1.0 / n;
  (* рисование осей *)
  moveto(min,min);
  lineto(min,max);
  lineto(max,max);
  (* построение прямой y = x *)
  moveto(min,max);
  lineto(max,min);
  x := 0;
  y := f(x, r, iterate);
  i := round(max - x * scale);
  j := round(max - y * scale);
  moveto(i,j)
  for m := 1 to n do
```

```
    begin
      x := x + delta;
      y := f(x, r, iterate);
      i := round(min + x * scale);
      j := round(max - y * scale);
      lineto(i,j)
    end
end;

procedure draw (x, r : real;
                iterate : integer);
const
  n = 100;          (* число итераций отображения *)
  min = 50;         (* используется для экранных координат *)
  max = 250;
  scale = 200;
var
  y, y0, x0 : real;
  i, j, i0, j0, m : integer;
begin
  x0 := x;
  y0 := 0.0;
  for m := 1 to n do
    begin
      y0 := f(x, r, iterate);
      i0 := round(min + scale * x);
      j := round(max - scale * y0);
      lineto(i,j)
      j := round(max - scale * y);
      lineto(i,j)
      i := round(min + scale * y);
      lineto(i,j)
      x0 := y;
      y0 := y;
      x := y
    end
end;
```

```
begin                          (* основная программа *)
  parameter(x, r, iterate);
  map(r, iterate);
  draw(x, r, iterate)
end.
```

ГЛАВА 8

```
program waves (input, output);
var
    A, v, lambda, dt, xmax, space, xscale, yscale : real;
    ntime : integer;

    procedure initial (var A, v, lambda, dt : real);
    begin
      write('v (см/с) = ');
      readln(v);
      A := 1.0;                        (* амплитуда волны *)
      lambda := 2.0 * 3.14159;         (* см *)
      write('время (с) между графиками = ');
      readln(dt)
    end;

    procedure screen (A, lambda, dt : real;
                      var xmax, space, xscale, yscale : real;
                      var ntime : integer);
    const
        x0 = 250;
        y0 = 300;
    var
        iaxes, ntick, itick, i, j, i0, j0 : integer;
        ymax, dx, dy, row, t : real;
    begin
      xmax := 6.0 * lambda;
      xscale := x0 / xmax;
      ntime := 5;              (* число рисуемых волн *)
      space := 0.5 * A;   (* расстояние между графиками *)
      ymax := ntime * (space + 2 * A);
      yscale := y0 / ymax;
      dx := lambda / 4.0;
```

```
  write('расстояние между делениями = ', dx);
  dy := A / 10.0;  (* высота деления *)
  row := A;
  ntick := round(xmax / dx);
  for iaxes := 1 to ntime do
      begin
        i := round(x0 + xscale * (-xmax));
        j := round(y0 - yscale * row);
        moveto(i,j)
        i := round(x0 + xscale * xmax);
        lineto(i,j)
        for itick := -ntick to ntick do
            begin
              i := round(x0 + itick * dx * xscale);
              j := round(y0 - yscale * row);
              moveto(i,j)
              j := round(y0 - yscale * (row + dy));
              lineto(i,j)
            end;
        row := row + space + 2.0 * A
      end
end;

function u (A, v, x, t : real) : real;
begin
  u := A * cos(x - v * t)
end;

procedure wavemotion (A, v, xmax, dt, space, xscale, yscale : real;
                      ntime : integer);
const
    x0 = 250;
    y0 = 300;
var
    row, dx, t, x : real;
    i, j, itime, ipoint, npoint : integer;
begin
  npoint := 200;
  dx := xmax / npoint;
  t := 0.0;
```

```
        row := A;                (* y-координата экрана для u = 0 *)
        for itime := 1 to ntime do
            begin
                t := t + dt;
                x := -dx * npoint;
                i := round(x0 + xscale * x);
                j := round(y0 - yscale * (u(A, v, x, t) + row));
                moveto(i,j)
                for ipoint := -npoint + 1 to npoint do
                    begin
                        x := dx * ipoint;
                        i := round(x0 + xscale * x);
                        j := round(y0 - yscale * (u(A, v, x, t) + row));
                        lineto(i,j)
                    end;
                row := row + space + 2.0 * A
            end
    end;

begin                            (* основная программа *)
    initial(A, v, lambda, dt);
    screen(A, lambda, dt, xmax, space, xscale, yscale, ntime);
    wavemotion(A, v, xmax, dt, space, xscale, yscale, ntime)
end.
```

ГЛАВА 9

```
program Laplace (input, output);
type
    matrix = array[1..100, 1..100] of real;
var
    V : matrix;
    nx, ny, iterations : integer;
    change : real;

    procedure assign (var V : matrix;
                      var nx, ny : integer;
                      var change : real);
```

```
var
    col, row : integer;
    V0 : real;
begin
  write('число ячеек по оси x = ');
  readln(nx);
  write('число ячеек по оси y = ');
  readln(ny);
  write('потенциал на границе прямоугольника в вольтах = ');
  readln(V0);
  write('повторять, пока относительная невязка не станет <= ');
  readln(change);
  change := change / 100.0;
  for col := 1 to nx do      (* фиксируем потенциал на границе *)
      begin
        V[col, 1] := V0;
        V[col, ny] := V0
      end;
  for row := 1 to ny do
      begin
        V[1, row] := V0;
        V[nx, row] := V0
      end;
  (* задаем начальные потенциалы внутренних ячеек *)
  for col := 2 to nx - 1 do
      for row := 2 to ny - 1 do
          V[col, row] := 0.9 * V0
end;

procedure outdat (var V : matrix;
                  nx, ny, iterations : integer);
var
    col, row : integer;
begin
  writeln;
  writeln('номер итерации = ', iterations);
  for row := 1 to ny do
      begin
```

```
            for col := 1 to nx do
                write(V[col, row] : 7 : 3);
            writeln
        end
    end;

    procedure iterate (var V : matrix;
                       var nx, ny, iterations : integer;
                       change : real);
    var
        Vave : matrix;
        diff, dmax : real;
        row, col : integer;
    begin
      iterations := 0;
      repeat
       dmax := 0.0;
       iterations := iterations + 1;
       for col := 2 to nx - 1 do
         for row := 2 to ny - 1 do
           begin
             Vave[col, row] := V[col + 1, row] + V[col - 1, row];
             Vave[col, row] := Vave[col, row] + V[col, row+1] + V[col, row-1]
             Vave[col, row] := 0.25 * Vave[col, row];
             (* вычисляем относительное изменение потенциала *)
             diff := abs((V[col, row] - Vave[col, row]) / Vave[col, row]);
             if diff > dmax then
                 dmax := diff
           end;
        for row := 2 to ny - 1 do
            for col := 2 to nx - 1 do
                V[col, row] := Vave[col, row];
        outdat(V, nx, ny, iterations);
      until dmax <= change (* повторяем итерации до требуемой точности *)
    end;

begin                          (* основная программа *)
    assign(V, nx, ny, change);
    iterate(V, nx, ny, iterations, change)
end.
```

ОГЛАВЛЕНИЕ