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 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 «улучшенного» 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

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-

нне 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, Даннеля Бен-Аврахама, Джона Дэвиса, Хьюга Де Уитта, Лиэу Дандон, Ферриндун Фамили, Джима Гивена, Джима Гантона, Мерилин Джименез, Габора Кальмана, Тома Кейза, Роберта Килмойера, Билла Клейна, Питера Клебана, Роджера Конна, Кристофера Лакин, Франсуа Лейвраза, Джона Махта, Джин Мазенко, Билла Мичалсона, Роберта Пелковитса, Джозефа Приста, Стена Райнака, Сидни Реднера, Питера Рейнолдса, Кун Ру, Дэвида Сторка, Орнола Валлса, Джералда Вишняка, Джорджа Вайсса, Питера Внешера 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-Гаиди, 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 изданне или пройтись по любой физической лаборатории, чтобы всюду встретить компьютеры. Говоря об использовании компьютеров в физнке, можно выделить четыре категорни:

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 бо́льшим, 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 лабораторным 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 бо́льшую 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).

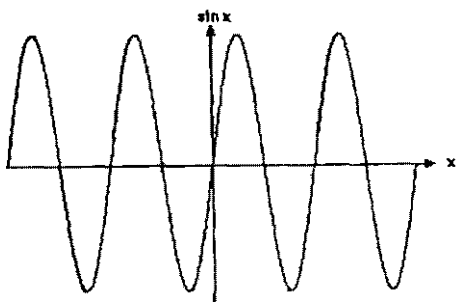

Рис. 1.1. График функции синус.

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. Кеменн 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 Кеменн 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 программы 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 — ие 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. Приводятся 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 постояниа, 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.

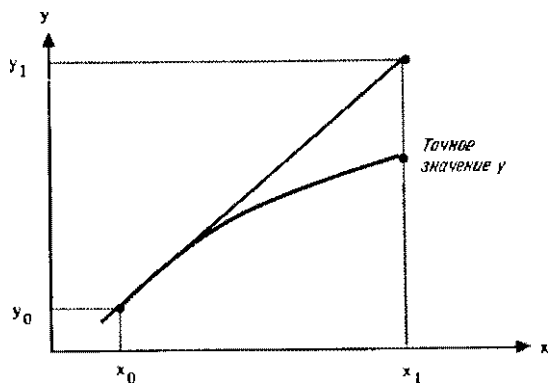

Рис. 2.1. Графическая интерпретация метода Эйлера. Наклон касательной вычисляется в начальной точке интервала. Приближению Эйлера и истинной функции соответствуют прямая и кривая.

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 печаянно 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) бесконечной 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 $T(t \to \infty) = T_s$.

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

цнфр, то можно предположнть, 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 нли происходит насыщение? Напишите программу для численного решения уравнения (2.8) методом Эйлера. Какой необходимо выбрать величину шага Δt, чтобы получнть решение с тремя правильными десятичными знаками в момент временн $t = 0.005$ 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 измененню 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 инструкцни на целом ряде типов компьютеров аналогичны, а поиимание принцнпов работы простых стандартных графических программ на одном компьютере будет достаточным для того, чтобы освоить компьютер другого типа нли легко одолеть другой графический язык. В дальнейшем мы рассмотрим основные графические команды языка True BASIC и разработаем программу для построенння на экране компьютера графика любой функции. Основные графическне команды для некоторых других языков и графических пакетов приводятся в конце первой части в приложенни по языкам Фортран и Паскаль.

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

раженне абсолютных значенй координат в координаты устройства, нлн пикселы, осуществляется в самом языке True BASIC. Первым делом прн использовании графического вывода в языке True BASIC необходимо задать днапазоны изменешня координат, в которых будут строиться графикн. Инструкция

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

определяет мининмальное и максимальное значення x (горизонтальной) и y (вертнкальной) координат и очнщает экран. Инструкцня

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

строит точку (x, y) в текущнх оконных координатах. Несколько *«базовых»* графическнх инструкцнй языка True BASIC приводятся в табл. 2.3. Некоторые другне команды будут рассматриваться в следующих главах по мере необходимостн.

ТАБЛИЦА 2.3. «Базовые» графнческие 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
```

Запустнте следующую программу и посмотрнте, как работают графнческие команды. Для этого вы могли бы ввестн в программу несколько инструкцнй 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$)
  INPUT prompt "минимум горизонтальной переменной? ": xmin
  INPUT prompt "максимум горизонтальной переменной? ": xmax
  INPUT prompt "минимум вертикальной переменной? ": ymin
  INPUT prompt "максимум вертикальной переменной? ": ymax
  INPUT prompt "название графика? ": title$
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 *силы*, 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.

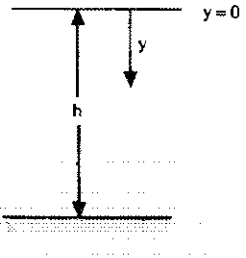

Рис. 3.1. Удобная система координат для задачи о теле, падающем с высоты h.

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-

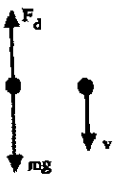

Рис. 3.2. Силы, действующие на падающее тело с учетом сопротивления воздуха.

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

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

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          ! (с)
  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                   ! вычисляется 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 «понаблюдать» 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 экспериментом. 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**.

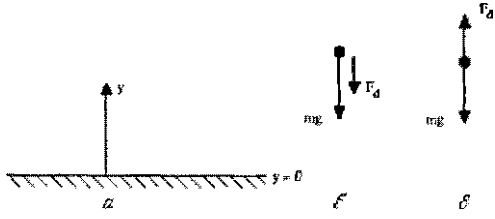

Рис. 3.3. Принятая в задаче 3.3 система координат с осью y, направленной вверх от земной поверхности (*а*). Силы, действующие на тело, движущееся вверх (*б*). Силы, действующие на тело, движущееся вниз (*в*).

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.

6. 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 $\mathrm{sgn}(x)$ имеет своим значением знак x. Или можно записать тормозящую силу в виде $F_d = -v*\mathrm{abs}(v)$. 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:

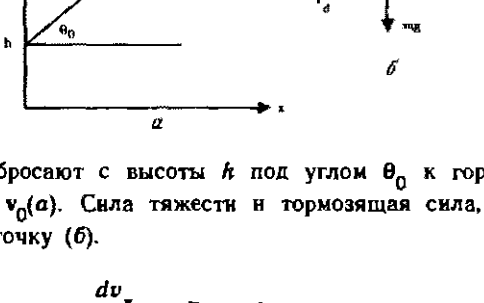

Рис. 3.4. Мяч бросают с высоты h под углом θ_0 к горизонту с начальной скоростью $\mathbf{v}_0$(*а*). Сила тяжести и тормозящая сила, действующие на материальную точку (*б*).

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

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

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

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

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

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 пренебрегните сначала сопротивленнем воздуха, чтобы можно было сравнить получаемые результаты с известными. Напрнмер, представьте, что шар бросают с поверхности Земли под углом θ_0 к горизонту с начальной скоростью $v_0 = 15$ м/с. Проварьируйте велнчину угла θ_0 и покажите, что максимальное расстояние, на которое улетнт 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, определяемой 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 $(x-y)$, 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

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

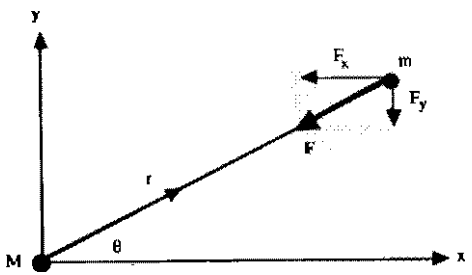

Рис. 4.1. Тело массой m движется под действием центральной силы F. Отметим, что выражения $\cos\theta = x/r$ и $\sin\theta = y/r$ позволяют записать уравнения движения в компонентах, что удобно для численного моделирования.

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-

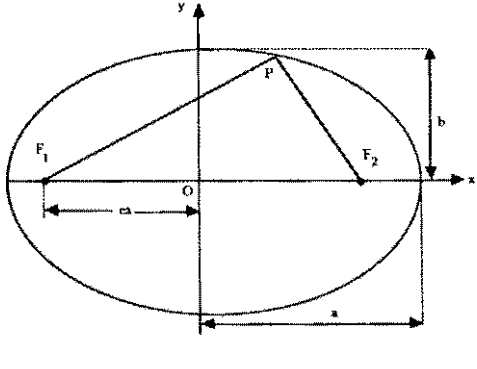

Рис. 4.2. Определение эллипса с помощью большой и малой полуосей a и b. Эксцентриситет e определен на рисунке. Начало декартовой системы координат O совпадает с центром эллипса.

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?

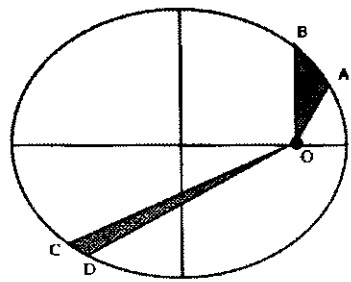

Рис. 4.3. Иллюстрация второго закона Кеплера равных площадей.

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 (э.е.) численное значение постоянной тяготения G равно

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

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

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

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

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

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 передаваемый 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-

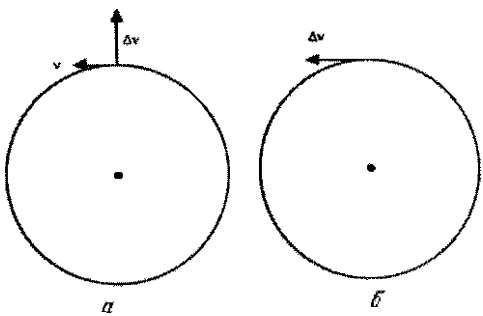

Рис. 4.4. Импульс силы приложен в радиальном (вертикальном) направлении (*а*). Импульс силы приложен в таигенциальном (горизонтальном) направлении (*б*).

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

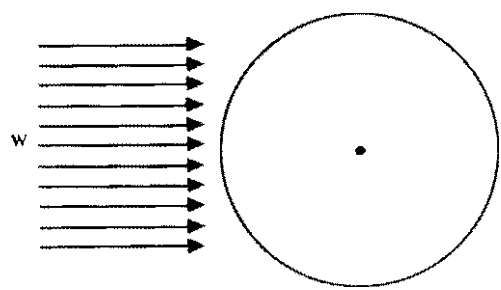

Рис. 4.5. Как изменится орбита под действием «солнечного ветра»?

движении можно записать в следующем виде:

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

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

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 Люрмана.)

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.

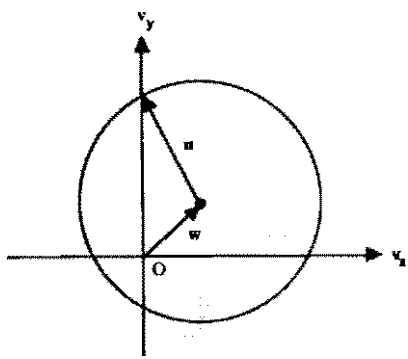

Рис. 4.6. Орбита материальной точки в пространстве скоростей. Вектор **w** направлен из начала координат пространства скоростей в центр круговой орбиты. Вектор **u** направлен из центра орбиты в точку с координатами (v_x, v_y).

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 изменениий 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