МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ
имени М.В.ЛОМОНОСОВА

Механико-математический факультет
Кафедра дифференциальной геометрии и приложений

Компьютерное моделирование и визуализация задач механики и геометрии

В.Л. Голо, Д.О. Синицын

Москва 2009 год

В.Л. Голо, Д.О. Синицын. Компьютерное моделирование и визуализация задач механики и геометрии. – М.: Изд-во ЦПИ при механико-математическом ф-те МГУ, 2009, – 120 стр.

Настоящее пособие представляет собой введение в методы компьютерного моделирования задач механики и геометрии. Его основной целью является научить читателя самостоятельно разрабатывать моделирующие программы и применять их в реальных условиях исследовательской работы. Подробно обсуждаются все этапы процесса моделирования на ряде примеров задач механики и геометрии. Рассматриваются вопросы визуализации изучаемых закономерностей.

Для студентов, аспирантов и научных работников, заинтересованных в применении компьютеров в исследованиях.

Рецензент: д. ф.-м. н., проф. А.А.Тужилин

Оглавление

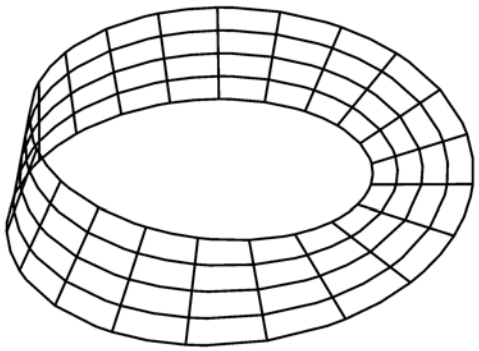
Предисловие

Настоящее учебное пособие основано на лекциях и практических занятиях по компьютерному моделированию задач механики и геометрии, проводимых авторами в течение ряда лет на кафедре дифференциальной геометрии и приложений механико-математического факультета МГУ. Целью занятий было научить применять компьютер в реальных условиях исследовательской работы.

Ввиду того, что занятия были ориентированы на студентов начиная с первого курса, материал подбирался исходя из условия достаточной простоты, не требующей сложного математического аппарата, однако, с другой стороны, отбирались наиболее содержательные задачи, затрагивающие фундаментальные вопросы механики.

Важнейшую роль в исследовании задач, в том числе, путем их компьтерного моделирования, играют способы представления полученных результатов и, прежде всего, способы графического изображения, визуализации исследуемых явлений. Это связано с тем, что многие закономерности в механике носят геометрический характер, связаны со свойствами определенных геометрических структур (ярким примером является движение планет, связанное с теорией конических сечений). С другой стороны, подходы механики и физики могут быть успешно применены к исследованию многих вопросов геометрии (например, соображения электростатики позволяют задавать поверхности с различными свойствами; в книге приводится пример получения двумерной сферы с ручками – одного из основных объектов топологической теории поверхностей). Таким образом, графическое представление информации позволяет проследить связи механики и геометрии, столь плодотворные для обеих наук.

В качестве основного способа компьютерного моделирования применяется написание собственных программ на алгоритмическом языке C++, дающее принципиальную универсальность возможностей и непосредственное понимание вычисли-



тельной природы используемых методов моделирования. При этом по ходу изложения даются необходимые сведения об используемых численных методах. Кроме того, объясняются приемы структурирования программ, необходимого для их успешного применения в исследованиях. Часть задач исследована с помощью программного пакета Mathematica, предоставляющего богатый набор удобных в использовании графических средств, а также широкие возможности по работе с формулами.

Для овладения материалом книги требуются начальные представления о механике Ньютона (на уровне средней школы), знание математического анализа и аналитической геометрии в объеме программы первого курса технических ВУЗов, понятие об обыкновенных дифференциальных уравнениях. Кроме того, предполагается первичное знакомство читателя с программированием и элементарными средствами языка C++.

Авторы пользуются приятной возможностью поблагодарить А.Т.Фоменко за внимание к этой работе.

Глава 1

Сила и масса

Задача механики состоит прежде всего в описании движения системы, т.е. нахождения зависимости характеристик системы от времени. Прежде всего надо решить, как задавать сами характеристики. Ответ на этот вопрос требует установления понимания физической сущности явления и далеко не всегда прост, поскольку включает оценку существенных и менее существенных деталей процесса. Здесь мы ограничимся разбором конкретных ситуаций и будем следовать принципу постепенного перехода от простого к сложному.

1.1. Второй закон Ньютона

Начнём с одномерного движения и будем считать, что размером нашей системы можно пренебречь и представлять её себе как точку, двигающуюся по прямой. В таком случае положение системы описывается одной координатой x, а её движение зависимостью $x(t)$ от времени t. Будем описывать внешнее воздействие на тело как силу f, а его внутреннюю характеристику – массой m. Второй закон Newton'а описывает движение системы согласно уравнению

$$\frac{dp}{dt} = f,$$

где p – количество движения, или импульс

$$p = m\,\frac{dx}{dt}.$$

Если внешнего воздействия нет, т.е. $f = 0$, то $p = const$. Будем в дальнейшем предполагать, что масса частицы постоянна. В таком случае имеем, что движение частицы описывается уравнением

$$x = x_0 + v\,t, \quad v = \frac{p}{m}$$

Таким образом, мы получаем частный случай общего закона инерции: тело находится в состоянии покоя или равномерного прямолинейного движения, пока на тело не действует сила.

Приведённые выше уравнения обощаются на трёхмерный случай

$$\frac{d}{dt}\vec{p} = \vec{f}, \quad \vec{p} = m\frac{d}{dt}\vec{x}$$

Формулировка закона инерции сохраняется.

Задача определения массы, входящей с формальной точки зрения как коэффициент во второй закон Newton'a, имеет первостепенное значение. Центральным вопросом здесь является проблема измерения массы. Обычная методика измерения посредством взвешивания предполагает совпадение инертной массы из второго закона и гравитационной массы, определяющей силу тяжести. В настоящее время, на основании результатов очень точных экспериментов, принято считать, что совпадение имеет место. Более подробное обсуждение этих вопросов требует серьёзного экскурса в общую теорию относительности. Мы примем фомальную точку зрения и будем рассматривать массу как коэффициент пропорциональности во втором закое Newton'a. В дальнейшем в качестве единицы массы принимается 1 $gramm$, времени 1 sec, длины 1 cm, т.е. мы будем пользоваться системой единиц CGS.

1.2. Применение закона сохранение импульса к теории удара

Законы сохранения энергии и импульса носят такой общий характер, что их часто воспринимают как основные законы природы, которые не нуждаются в выводе из каких-либо других предположений. В действительности, они следуют из свойств симметрии пространства-времени. Мы рассмотрим здесь конкретный пример одномерного движения в отсутствие трения и примем закон сохранения энергии и импульса без доказательства. Наша цель – проиллюстрировать математическую мощь этих законов сохранения на физическом примере, имеющем и самостоятельный интерес.

1.2.1. Одномерный случай

Рассмотрим упругое соударение двух материальных точек. Будем исходить из законов сохранения импульса и энергии

$$P = p_1 + p_2, \qquad E = \frac{p_1^2}{2m_1} + \frac{p_2^2}{2m_2}$$

Решая эту систему уравнений, получаем решение

$$\begin{aligned} p_1 &= \frac{m_1}{m_1+m_2}\,P \pm \sqrt{\frac{2m_1m_2}{m_1 1+m_2}\,\left(E - \frac{P^2}{m_1+m_2}\right)} \\ p_1 &= \frac{m_2}{m_1+m_2}\,P \mp \sqrt{\frac{2m_1m_2}{m_1+m_2}\,\left(E - \frac{P^2}{m_1+m_2}\right)} \end{aligned}$$

которое соответствует двум парам импульсов рассматриваемых материальных точек – до и после удара. Отметим особо частный случай частиц равной массы $m_1 = m_2 = m$. Приведённая выше система уравнений геометрически соответствует пересечению прямой, параллельной биссектрисе второго и четвёртого координатного угла и окружности. Отсюда заключаем, что в результате удара частицы будут обмениваться импульсами, т.е. значения последних до и после удара, p_1', p_2' и p_1'', p_2'' связаны соотношением

$$p_1'' = p_2', \qquad p_2'' = p_1'$$

Отметим, что из этих соотношений следует, что удару соответствует преобразовние плоскости импульсов p_1, p_2, сохраняющее площадь.

1.2.2. Столкновение шаров

Столкновение шаров в трёхмерном пространстве связано с любопытными математическими явлениями. Будем предполагать, что шары равной массы и равного радиуса и что закон сохранения энергии и импульса выполняются. Это означает, что в процессе удара не происходит изменения внутренней энергии шара. В общем случае это не так. Представим себе, например, что в результате удара шар начинает колебаться, что может иметь место в случае шара, представляющего собой резиновую

оболочку, наполненную газом. Тогда энергия поступательного движения не будет сохраняться, поскольку при ударе часть её перейдет в энергию колебаний шара. Таким образом, у нас речь пойдёт скорее о шарах типа бильярдных. Строго говоря, у последних также будут возникать внутренние колебания как последствия удара, но их можно считать достаточно малыми. Также не будем учитывать возможные вращения шаров и их изменение до и после удара. Для наглядности запишем законы сохранения импульса и энергии посредством скоростей центров шаров

$$\vec{v}_1'' + \vec{v}_2'' = \vec{v}_1' + \vec{v}_2' \tag{1.1}$$

$$\vec{v}_1''^{\,2} + \vec{v}_2''^{\,2} = \vec{v}_1'^{\,2} + \vec{v}_2'^{\,2} \tag{1.2}$$

Считая скорости шаров до удара заданными, мы имеем 4 уравнения для 6 неизвестных – компонент скоростей шаров после удара. Таким образом, решение будет зависеть от двух произвольных постоянных. Мы всегда можем представить векторы $\vec{v}_1''$, $\vec{v}_2''$ в виде

$$\vec{v}_1'' = +\vec{Y}, \quad \vec{v}_2'' = \vec{v}_2' + \vec{Z}$$

Из уравнения (1.1) следует, что $\vec{Z} = -\vec{Y}$, причём направление вектора $\vec{Y}$ остаётся произвольно. Модуль $\vec{Y}$ найдём, воспользовавшись уравнением (1.2). Для этого положим $\vec{Y} = x\,\vec{W}$, где $\vec{W}$ – единичный вектор. Подставив выражения

$$\vec{v}_1'' = \vec{v}_1' - x\,\vec{W}, \quad \vec{v}_2'' = \vec{v}_2' - x\,\vec{W}$$

в уравнение (1.2), получим

$$\vec{v}_1'' = \vec{v}_1' - \frac{1}{2}\,(\vec{V}\cdot\vec{W})\,\vec{W}, \quad \vec{v}_2'' = \vec{v}_2' + \frac{1}{2}\,(\vec{V}\cdot\vec{W})\,\vec{W}$$

где вектор $\vec{V}$ есть вектор относительной скорости $\vec{V} = \vec{v}_1' - \vec{v}_2'$. Значение "произвольного"вектора $\vec{W}$ в действительности определяется характером удара. Если предположить, что соприкосновение шаров в момент удара будет точечным, то расположение шаров в момент удара будет описываться вектором, направленным по их линии центров. Если предположить, что кажется вполне естественным, что передаваемый при этом импульс

дожен быть направлен по общей нормали к сферам – поверхностям шаров, то, учитывая, что он пропорционален относительной скорости $\vec{V}$, получаем, что единичный вектор $\vec{W}$ есть вектор направления линии центров. Более подробное описание столкновения шаров потребовало бы детального исследования характера столкновения.

1.3. Изменение импульса в интервале времени

Ситуация, с которой мы сталкиваемя в теории удара, весьма характерна. Суть её в том, что мы не знаем детали явления, и ввиду этого прибегаем к законам сохранения. В этой связи разберём одномерную задачу разд. 1.2.1.

Будем предполагать, что удар происходит в течение интервала времени τ, на протяжении которого на первую точку действует сила $f(t)$, а на вторую сила $-f(t)$, согласно 3-му закону Newton'a, *действие равно противодействию*. Далее предположим, что $f(t) = 0,\ t < 0,\ t \geq \tau$, т.е. до и после удара точки двигаются свободно. В течение процесса соударения движение описывается уравнениями

$$\dot{p}_1 = f(t), \quad \dot{p}_2 = -f(t)$$

Откуда следует сохранение импульса $p_1 + p_2 = const$. Усмотрим также, что из сделанных предположений следует сохранение энергии. Для этого умножим уравнения движение на p_1, p_2 соответственно и сложим, получим

$$\frac{p_1\,\dot{p}_2}{m_1} + \frac{p_2\,\dot{p}_2}{m_2} = f(t)\,\left(\frac{p_1}{m_1} - \frac{p_2}{m_2}\right)$$

В течение удара обе точки двигаются вместе, т.е.

$$\frac{p_1}{m_1} = \frac{p_2}{m_2}$$

Откуда следует сохранение энергии. Оба закона сохранения оказались следствием 3-го закона Newton'a.

Примером ситуации, в которой сохраняется энергия, но не импульс, является столкновение свободной частицы с осциллятором.

Пусть имеются две материальные точки m_1, m_2; первая движется свободно, вторая находится в поле силы, подчиняющейся закону Гука. В течение удара на них действуют силы $f(t)$ и $-f(t)$ соответственно. Таким образом, уравнения движения имеют вид

$$\dot{p}_1 = f(t), \quad \dot{p}_2 = -f(t) - k\, x_2$$

Ясно, что импульс $p_1 + p_2$ сохраняться не будет. Однако будет сохраняться энергия, как это явствует из уравнения, непосредственно вытекающего из уравнений движения, приведённых выше:

$$\frac{\dot{p}_1 p_1}{m_1} + \frac{\dot{p}_2 p_2}{m_2} = f(t)\, \left(\frac{p_1}{m_1} - \frac{p_2}{m_2} \right)$$

В момент соударения частицы двигаются с равными скоростями, таким образом

$$\frac{p_1}{m_1} - \frac{p_2}{m_2} = v_1 - v_2 = 0$$

поскольку в отсутствие столкновения $f(t) = 0$. Интегрируя приведённое выше уравнение, получаем закон сохранения энергии

$$\frac{m_1\, v_1^2}{2} + \frac{m_1\, v_1^2}{2} + \frac{k}{2}\, x_2^2 = const$$

Мы видим, что даже в очень простых ситуациях закон сохранения импульса перестаёт работать. Тем не менее, из второго закона Ньютона вытекает закон изменения импульса свободной частицы за некоторый интервал времени Δt

$$\Delta\, p = \int\limits_{\Delta t} f(t)$$

Это очень общее соотношение, эквивалентное второму закону, в интегральной форме. Мы применим его к вопросу, формально относящемуся к явлениям, выходящим далеко за пределы обычной механики, в целях иллюстрации мощи методов аналитической механики.

1.3.1. Гипотеза статистического описания

Рассмотрим систему очень большого числа $N \gg 1$ одинаковых частиц. Уследить за изменением состояния индивидуальной частицы мы не сможем. Поэтому в случае очень большого числа частиц мы будем ориентироваться на статистическое описание и ограничимся усреднёнными характеристиками динамики частиц.

Физическая картина, лежащая в основе последующих рассуждений, основывается на концепции идеального газа. Мы будем понимать под последним совокупность $N \gg 1$ частиц, взаимодействие между которыми достаточно мало, чтобы им можно было пренебречь, рассматривая взаимодействия между отдельными индивидуальными частицами, т.е. можно пренебречь индивидуальными актами столкновения, но тем не менее оно достаточно, чтобы можно было говорить о некоторой средней квадратичной скорости, $\overline{v^2}$.

1.3.2. Вывод уравнения состояния идеального газа

Рассмотрим задачу о вычислении давления идеального газа на стенку сосуда. Следует отметить, что это многочастичная задача, и таким образом она отличается от того, что у нас было до сих пор. Вместе с тем концепция идеального газа предполагает, что парным взаимодействием частиц можно пренебречь и описывать динамику индивидуальной частицы, т.е. рассматривать одночастичную задачу. Затем из полученных результатов будут выводиться свойства системы многих частиц с помощью статистических методов.

Будем исходить из того, что газ оказывает давление на стенку сосуда за счёт столкновений молекул газа со стенкой. При таком подходе воздействие молекул на стенку будет носить характер случайных толчков. Дадим описание этого процесса на временах T много больших молекулярных, т.е. на временах много больших, чем характерные времена столкновений молекул со стенкой, τ. Столкновениями самих молекул мы пренебрегаем. В течение столкновения i-й молекулы со стенкой на последнюю действует сила $f_i(t)$, отличная от нуля только на временном интервале $t_i < t < t_i + \tau$. В течение интервала времени $0 < t < T$

за это время произойдёт n столкновений молекул со стенкой. Среднее суммарное воздействие будет даваться формулой

$$\overline{f} = \frac{1}{T} \int\limits_0^T dt \sum_{i=1}^n f_i(t) = \sum_{i=1}^n \frac{1}{T} \int\limits_0^T dt\, f_i(t) \tag{1.3}$$

Существенно, что в результате индивидуального i-го столкновения некоторой молекулы со стенкой изменение импульса, как это следует из второго закона, даётся формулой

$$p(T) - p(0) = \int\limits_0^T f_i(t)\, dt$$

откуда явствует, что усреднённое воздействие на стенку сосуда за счёт столкновений её с молекулами представимо в виде

$$\overline{f} = \frac{1}{T} \sum_{i=1}^n (p_{i,fin} - p_{i,init}) \tag{1.4}$$

Для того, чтобы найти давление, нам нужно рассмотреть воздействие на единицу площади стенки и учесть то обстоятельство, что столкновение со стенкой может быть под углом.

Существенно, что разность импульсов до и после столкновения в реальной ситуации нам не известна. Сделаем упрощающие (и физически оправданные) предположения. Поскольку стенка массивна, а молекула мала, мы предположим, что в результате столкновения скорость стенки остаётся равной нулю, т.е. мы предполагаем, что стенка покоится, ввиду чего имеем $p_{init} = -p_{fin}$ и для их разности получаем

$$p_{init} - p_{fin} = 2mv$$

где v – скорость по нормали к стенке. Согласно формуле (1.4) мы имеем

$$\overline{f} = \frac{1}{T} \sum_{i=1}^n 2m_i\, v_i$$

где индекс i занумеровывает акты отдельных столкновений. Правую часть можно переписать в виде суммы вкладов, даваемых столкновениями отдельных частиц, занумерованных индексом j,

$$\overline{f} = \frac{1}{T} \sum_{j=1}^{N} K_j \cdot 2m_j\, v_j$$

Здесь K_j есть число столкновений j-й молекулы со стенкой за время T; v_j обозначает нормальную к стенке компоненту скорости частицы. Величину K_j можно вычислить, принимая во внимание, что величина нормальной компонены скорости не меняется в результате столкновения. Если размер сосуда по вертикали L_z, то время пролёта от верхнего до нижнего основания равно

$$\tau_0 = \frac{L_z}{v_j}$$

Следовательно, время отскока от нижнего основания и возвращения равно $2\,\tau_0$. Отсюда число столкновений j-й частицы за интервал времени T равно

$$K_j = \frac{T\, v_j}{2\tau_0}$$

Таким образом,

$$\overline{f} = \sum_{j=1}^{N} \frac{m_j\, v_j^2}{L_z}$$

Чтобы получить давление, т.е. силу на единицу площади, надо разделить силу на площадь основания $L_x\, L_y$. Учитывая, что объём сосуда равен $V = L_x\, L_y\, L_z$, получаем

$$P = \frac{\overline{f}}{L_x L_y} = \frac{1}{V} \sum_{j=1}^{N} m_j\, v_j^2 = \frac{2}{V} \sum_{j=1}^{N} \frac{m_j\, v_j^2}{2} \qquad (1.5)$$

Аналогичные формулы получаем для направлений x и y, в результате имеем

$$3P = \frac{2}{V} \sum_{j=1}^{N} \left(\frac{m_j\, v_{x,j}^2}{2} + \frac{m_j\, v_{y,j}^2}{2} + \frac{m_j\, v_{z,j}^2}{2} \right) = \frac{2}{V} \sum_{j=1}^{N} \frac{m_j\, \vec{v}_j^{\,2}}{2}$$

Далее, на основании закона больших чисел, положим

$$\overline{v^2} = \frac{1}{N} \sum_{j=1}^{N} \vec{v}_j^2$$

Получаем

$$PV = \frac{2}{3} N \frac{m\overline{v^2}}{2}$$

Таким образом, в правой части появилась средняя кинетическая энергия молекулы газа. Основная гипотеза кинетической теории газов связывает её с температурой, которая вводится в термодинамике, посредством соотношения

$$\frac{m\overline{v^2}}{2} = \frac{3}{2} k_B T$$

где k_B – постоянная Больцмана. Вспомним, что N – число молекул в данном объёме газа. В случае, когда имеется один грамм-моль газа, – это число Авогадро $N_A \approx 6 \cdot 10^{23}$. В результате можем записать уравнение состояния идеального газа в обычном виде

$$PV = RT, \quad R = N_A k_B \tag{1.6}$$

где R называется универсальной газовой постоянной.

1.3.3. Перенос импульса в газе. Формула Максвелла

Теперь обратимся к тому случаю, когда газ является неидеальным, т.е. частицы могут сталкиваться между собой. Оказывается, что и в этом случае концепция импульса частицы позволяет получить, при определённых условиях, полезные выводы о динамике уже многочастичной системы, которой является газ.

Необходимо сделать некоторые существенные предположения о характере течения жидкости. Конечно, согласно молекулярно-кинетическим представлениям, движение отдельных молекул в жидкости хаотично и во многом определяется их столкновениями друг с другом. Основным предположением, без которого не может быть и речи о каких-либо теоретических

конструкциях для описания газа, является представление о существовании некоторых усреднённых характеристик, т.е. возможности статистического описания, о котором говорилось выше.

Рассмотрим наиболее регулярный тип движения газа, когда течение на масштабе много большем, чем межмолекулярное расстояние, можно представить себе как бы происходящим послойно, т.е. происходящим по слоям, которые проскальзывают друг относительно друга. Такой тип течения называется ламинарным. Существенно, что отдельные молекулы газа могут всё же мигрировать из слоя в слой, при этом их импульсы будут меняться за счёт столкновения молекул между собой. Изменение импульса можно представить происходящим в результате внешнего воздействия или силы на отдельную молекулу. Мы, таким образом, не хотим вникать в сам характер межмолекулярного столкновительного взаимодействия. Однако, суммируя по всем молекулам, участвующим в переходе из некоторого слоя в соседний, мы получим результирующую силу, действующую между слоями, которая будет определять взаимодействие между ними, или трение слоёв газа друг о друга. Можно говорить, что между слоями возникает напряжение, которое соответствует наличию *вязкого трения*, соответствующего наличию переноса импульса от слоя к слою.

Описанный процесс переноса будет определяться тем расстоянием, на которое может свободно переместиться молекула, т.е. при перемещении на которое ещё может не произойти столкновение. Это расстояние называется длиной свободного пробега. Если мы будем рассматривать перемещения на расстояния большие длины свободного пробега, то нельзя уже говорить о переносе импульса из данного слоя в выбранный слой, потому что столкновения молекулы в промежуточных слоях приведут к тому, что часть импульса будет как бы утерена по пути.

Этим соображениям можно придать более количественную форму. Начнём с оценки величины свободного пробега. Для этого прежде всего заметим, что длина свободного пробега определяется размером молекулы, её радиусом a и их числом в единице объёма n. Мы предполагаем, что молеула имеет форму шара. Найдём длину свободного пробега l как функцию a и n. Чтобы получить грубую оценку, рассмотрим упрощённую ситуацию,

когда молекула движется поступательно со скоростью v, а прочие молекулы покоятся. Тогда столкновение может произойти только с молекулами, которые находятся на расстоянии a от отрезка прямой, описываемого центром молекулы, т.е. только с молекулами, которые попадут в соответствующий цилиндр с площадью основания πa^2 и высотой vt, где t – интервал времени, на котором рассматривается движение молекулы. Число таких молекул будет $K = n\pi a^2 vt$. Отсюда получаем, что среднее расстояние между двумя последовательными встречами, т.е. длина свободного пробега, будет

$$l = \frac{vt}{K} = \frac{1}{\pi a^2 n}$$

Приведённая оценка длины свободного пробега, конечно, достаточно грубая. Более точное рассуждение даёт коэффициент $1/\sqrt{2}$ в правой части.

Обратимся к вязкому трению и постараемся и его описать количественно. Здесь теоретические соображения покоятся на эмпирическом законе, восходящем ещё к Ньютону [1)]. А именно, сила вязкого трения f на единицу площади между соседними слоями в ламинарном потоке жидкости или газа выражается через градиент скорости v слоёв по нормали к слоям согласно формуле

$$f = \mu \frac{dv}{dz} \tag{1.7}$$

где μ называется коэффициентом вязкости. Указанную формулу можно проиллюстрировать следующим образом. Рассмотрим течение газа между двумя параллельными пластинками. Пусть расстояние между пластинками равно H. Нижняя пластинка закреплена и неподвижна, в то время как верхняя движется с постоянной скоростью U параллельно пластинкам. Предположим, что газ прилипает к пластинкам без проскальзывания, т.е. скорость газа на пластинках равняется скорости

[1)] Внимательный "анализ текстов т.е. учебников и солидных монографий, показывает, что все имеющиеся в настоящее время выводы уравнений динамики *вязкой жидкости* так или иначе основаны на предположении, что уравнение (1.7) имеет место. Мы имеем в виду именно жидкость, а не слабо неидеальный газ, для которого можно вывести уравнения гидродинамики, включая вязкие члены, из микроскопических соображений.

пластинок, в то время как слои газа между пластинками скользят друг относительно друга, испытывая взаимное трение [2)]. Оказывается, что в рассматриваемом случае профиль скоростей слоёв – линеен и даётся формулой

$$v = U\,\frac{z}{H}$$

К верхней и нижней пластинкам приложены противоположно направленные силы, каждая из них, отнесённая к единице площади, будет равна

$$f = \mu\,\frac{U}{H}$$

в соответствии с формулой (1.7).

Уравнение (1.7) является определением коэффициента вязкости μ. Получим для него выражение через микроскопические характеристики газа, пользуясь картиной переноса импульса между слоями газа в ламинарном течении. Рассмотрим два параллельных слоя жидкости на расстоянии dz, порядка длины свободного пробега l, друг от друга, движущихся со скоростями v и $v + dv$ соответственно. Условие, что dz – порядка длины свободного пробега, означает, что при переходе из слоя в слой молекула будет испытывать одно единственное столкновение с молекулой из другого слоя.

Вычислим изменение импульса слоёв за счёт перехода молекулы из слоя в слой через элементарную площадку в единицу времени. Пусть n есть число молекул в единице объёма, u – средняя скорость молекул. Предположим, что по каждому направлению x, y, z движется примерно одинаковое число молекул [1)], $n/3$. Кроме того, предположим, что перенос из слоя в слой симметричен, и, таким образом, в каждый из слоёв будет поступать половина из числа возможных молекул, т.е. $n/6$. В каждый из слоёв через площадку величины dA в единицу времени попадёт

$$\frac{1}{6}\,n\,V\,dA$$

[2)] Указанный режим вполне реалистичен; он удовлетворяет общим уравнениям динамики вязкой жидкости или газа и может быть осуществлён экспериментально.

[1)] Это довольно естественно и соответствует предполагаемой изотропии пространства.

молекул. Изменение импульса совокупности молекул, выходящих из слоя, где средняя скорость равна $V + dV$, равно $-1/6nVdA(mdV)$, аналогично для молекул, идущих из слоя, где средняя скорость равна V, изменение импульса равно $1/6nVdA(mdV)$, т.е. суммарное изменение импульса в едиицу времени

$$\frac{1}{3}\, n\, dA\, (mdV)$$

и соответственно сила, с которой один слой действует на другой,

$$dF = \frac{1}{3}\, nV\, m\, dV$$

что даёт вязкую силу, или касательное напряжение, на единицу площади

$$f = \frac{dF}{dA} = \frac{1}{3}\, nV\, m\, dV = \frac{1}{3}\, nV\, m\, \frac{dV}{dz}\, dz$$

Но, как мы предположили, $dz \approx l$, и согласно (1.7) мы получаем

$$\mu = \frac{1}{3}\, nV\, m\, l \tag{1.8}$$

Заметим, что $n\, m$ есть масса всех частиц в одном cm^3, т.е. плотность ρ. Таким образом, принимая во внимание полученную ранее формулу для свободного пробега, получаем формулу Максвелла

$$\mu = \frac{m\, V}{3\pi\, a^2}$$

для коэффициента вязкости. Из этой формулы получаем неожиданное следствие: вязкость не зависит от плотности и давления газа; с ростом температуры вязкость возрастает [1)].

1.3.4. Ламинарное течение. Формула Пуазейля

Рассмотрим ещё в качестве примера, как очень простые соображения позволяют получить важный физический результат,

[1)] В своё время это заключение вызвало оживлённую дискуссию. Понадобился авторитет Максвелла, чтобы убедить научную общественность в его правомерности.

течение Пуазейля. Пусть имеется цилиндрическая труба радиуса R и длины L. По трубе с постоянной скоростью V течёт несжимаемая вязкая жидкость за счёт перепада давления на концах трубы $P_1,\ P_2$. Движение, таким образом, можно считать установившимся, т.е. давление скомпенсировано вязкими силами, действующими на внутренней поверхности трубы. Существенно, что в соответствии с законом Паскаля давление по сечению трубы, соответствующему некоторому расстоянию от концов, постоянно. Мысленно выделим в трубе цилиндр радиуса r, коаксиальный трубе, т.е. с осью, совпадающей с осью трубы. Течение жидкости в выделенном цилиндре – стационарно, что достигается за счёт того, что перепад давлений на торцах цилиндра $P_1 - P_2$ уравновешивается силами трения поверхностной части цилиндра о прилегающую к ней остальную часть жидкости в трубе, т.е. имеет место баланс сил

$$(P_1 - P_2)\,\pi\,R^2 = 2\pi RL\ f(r)$$

где слева стоит разность суммарных давлений на торцы цилиндра, а справа – сила полного вязкого трения на поверхности цилиндра. Таким образом, имеем соотношение

$$f(r) = \frac{P_1 - P_2}{2L}\ r$$

Соответственно, если рассмотреть цилиндр радиуса R, т.е. всю трубу, то получим

$$f(R) = \frac{P_1 - P_2}{2L}\ R$$

Это соотношение играет роль граничного условия. Сила вязкого трения на единицу площади $f(r)$ согласно (1.7) равна

$$f(r) = -\mu\,\frac{dv}{dr}$$

Окончательно получаем дифференциальное уравнение для v – скорости жидкости на расстоянии r от оси трубы

$$\frac{dv}{dr} = -\ \frac{P_1 - P_2}{L}\ \frac{r}{2\mu}$$

Учитывая условие прилипания, которое в данном случае имеет вид

$$v(r = R) = 0 \tag{1.9}$$

получаем профиль скоростей в трубе

$$v = \frac{P_1 - P_2}{L} \frac{R^2 - r^2}{4\,\mu}$$

Интегрируя это уравнение, получаем выражение для потока жидкости по трубе

$$Q = 2\pi \int_0^R vrdr = \frac{P_1 - P_2}{L} \frac{\pi r^4}{8\mu}.$$

Эта формула называется формулой Пуазейля.

Заметим в заключение, что условие прилипания совсем не очевидно. Его выполнимость тщательно исследовалась в работах XIX века, и в экспериментах того времени оно было подтверждено. С этой целью замерялся расход жидкости, текущей под постоянным давлением по капилляру. Полученные результаты очень хорошо согласовывались с формулой Пуазейля, из чего делался вывод, что прилипание есть. Но в конце XX века появилась новая ветвь гидродинамики – микрогидродинамика, изучающая, в частности, течения в сверхтонких капиллярах. Было обнаружено, что, хотя основные уравнения течения вязкой жидкости продолжают давать верное описание течения, граничное условие прилипания не выполяется. Аналогичное явление было обнаружено Кеезомом (W.H.Keesom) ещё в 20-х годах прошлого века, для случая течения разреженного газа по капиллярам.

1.4. Задача для компьютерного моделирования: одномерный газ из одной частицы под поршнем при постоянном давлении

1.4.1. Мотивировки и постановка задачи

В качестве иллюстрации процессов, связанных с движениями молекул, которые происходят в газе, рассмотрим наиболее

упрощенную систему, в которой присутствуют подобные движения. Это система, состоящая из одной частицы, которая движется вдоль прямой. Для этой системы мы хотим рассмотреть динамику в режиме, похожем на изобарический режим для газа в сосуде: мы помещаем частицу между неподвижной стенкой и поршнем, на который действует постоянная сила. Удобно представлять себе прямую, вдоль которой происходит движение, расположенной вертикально, а силу, действующую на поршень, – силой тяжести (частицу считаем не подверженной гравитации).

Таким образом, частица движется равномерно между каждыми двумя последовательными столкновениями с поршнем либо с нижней стенкой сосуда. Поршень в те же промежутки совершает равноускоренное движение. В моменты столкновений скорости тел скачкообразно меняются. Столкновение со стенкой описывается в приближении бесконечности ее массы, а с поршнем – по точным уравнениям, следующим из законов сохранения импульса и энергии.

Проводя аналогию с обычным газом, можно считать силу тяжести, действующую на поршень, аналогом давления, высоту поршня (то есть величину участка прямой, предоставленного частице) – аналогом объма, а кинетическую энергию частицы – аналогом температуры. Поэтому встает вопрос, существует ли для этой системы некоторый аналог уравнения состояния идеального газа. Иными словами, если состояние системы удовлетворяет соотношению, аналогичному (1.6), то будет ли такое состояние иметь какие-то особые свойства? Для изучения этого вопроса проведем компьютерное моделирование нашей системы.

1.4.2. Программная реализация

Программа, моделирующая некоторый реальный объект, должна выполнять следующие две основные функции:

А) создавать в памяти компьютера структуру, имитирующую изучаемый реальный объект;

Б) вычислять интересующие нас характеристики этой модель-

ной структуры и отображать их в виде, удобном для восприятия человеком.

Заметим, что традиционное математическое моделирование (аналитическими методами) также можно представить в соответствии с этой схемой. При этом в случае именно компьютерного моделирования существенно возрастает роль второй функции, так как здесь есть принципиальная возможность (по крайней мере для не слишком громоздких моделей) вычислить любую характеристику, но для того, чтобы знание ее привело к лучшему пониманию сути моделируемого явления, необходимо весьма серьезно подойти к ее выбору, а также к способу отображения.

Приступим к реализации задачи А) для нашей системы. Поскольку, как было сказано в предыдущем пункте, движение тел состоит из участков между столкновениями, в течение которых координаты тел изменяются по известным формулам соответственно равномерного и равноускоренного движения, то имеет смысл организовать моделирование этого движения в виде последовательности пересчетов координат тел от момента после одного столкновения к моменту после следующего столкновения. Такой пересчет будет включать расчет движения до следующего столкновения и изменения скоростей в момент столкновения.

Последовательность этих пересчетов можно осуществить в цикле следующим образом[1)]:

```
// рассчитываем движение на промежутке,
// содержащем 500 столкновений
  for (int n = 1; n <= 500; n++)
  {
    Evolution_until_after_next_collision(x, v,
                                         X, V, t);
    // . . .   (обработка полученных данных)
  }
```

Здесь переменная `n` содержит номер текущего участка движения между столкновениями, 500 – число моделируемых столкновений, `x`, `v` – координата и скорость части-

[1)] Полный текст программы см. на стр. 32.

цы, `X`, `V` – координата и скорость поршня[2)], `t` – время, `Evolution_until_after_next_collision` – функция, осуществляющая пересчет координат и скоростей до момента сразу после очередного столкновения. Ей передаются значения переменных после предыдущего повторения цикла, а она изменяет их, приводя к новому состоянию. Знаком `"// . . ."` помечен участок, обрабатывающий вычисленные значения переменных (для нахождения интересующих нас характеристик системы, то есть реализации пункта Б) в изложенной выше схеме), который мы напишем позднее.

Конструкцию, подобную указанной, часто называют основным циклом вычислений. Она возникает во многих программах, которые моделируют процессы, развивающиеся во времени. Такие программы включают один или, в более сложных случаях, несколько основных циклов, отвечающих определенным режимам динамики. В каждом таком цикле осуществляется повторение однотипного набора операций, реализующих переход системы к следующему шагу по времени.

Теперь приступим к реализации основной функции цикла для нашего случая `Evolution_until_after_next_collision`. Как нам известно, частица движется между столкновениями равномерно, поэтому получаем для ее координаты и скорости выражения:

$$x(\tau) = x_0 + v_0\tau,$$
$$v(\tau) = v_0.$$

Здесь время τ отсчитывается от начала данного участка равномерного движения[1)], x_0, v_0 – координаты и скорость частицы на момент начала участка. Для поршня имеем равноускоренное движение с ускорением $-g$, откуда получаем:

$$X(\tau) = X_0 + V_0\tau - \frac{g\tau^2}{2},$$
$$V(\tau) = V_0 - g\tau.$$

Необходимо определить, будет ли очередное столкновение частицы столкновением с нижней стенкой сосуда или с поршнем.

[2)] Здесь мы считаем частицу материальной точкой, а в качестве координаты поршня рассматриваем высоту его нижней поверхности.

[1)] В отличие от времени t, отсчитываемого от начала движения в целом; аналогичные обозначения используются в программе.

Время возможного столкновения со стенкой вычисляется из условия $x(\tau) = 0$, откуда оно равно:

$$\tau_{bottom} = -\frac{x_0}{v_0}.$$

Очевидно, если это время меньше нуля, то столкновение со стенкой было в прошлом, частица движется вверх, и следующим будет столкновение с поршнем. Время возможного столкновения с поршнем рассчитывается из условия $x(\tau) = X(\tau)$, то есть

$$x_0 + v_0\tau = X_0 + V_0\tau - \frac{g\tau^2}{2},$$

или, эквивалентно,

$$\frac{g\tau^2}{2} + (v_0 - V_0)\tau + x_0 - X_0 = 0.$$

Так как в нашей задаче частица всегда находится ниже поршня, то имеем $x_0 - X_0 \leqslant 0$. Поэтому (а также потому что $g > 0$) это квадратное уравнение обязательно имеет корни, и они имеют разные знаки. Следовательно, в качестве времени будущего столкновения мы можем взять только больший из них, то есть тот, в котором выбирается плюс перед корнем из дискриминанта:

$$\tau_{piston} = \frac{-(v_0 - V_0) + \sqrt{(v_0 - V_0)^2 - 2g(x_0 - X_0)}}{g}.$$

Используя результаты проведенных вычислений, пишем начальную часть нашей функции, вычисляющую τ_{bottom} и τ_{piston}:

```
// Функция, пересчитывающая координаты частицы и поршня
// к моменту сразу после следующего столкновения
void Evolution_until_after_next_collision(double& x,
double& v, double& X, double& V, double& t)
{
  double tau_bottom = - x / v;
  // время до возможного столкновения с дном

  double D = (v-V) * (v-V) - 2. * g * (x - X);
  double tau_piston = (V - v + sqrtl(D)) / g;
  // время до возможного столкновения с поршнем
  // . . .
```

Как было замечено, если $v_0 > 0$ (а значит, $\tau_{bottom} < 0$), то частица летит вверх, и последует столкновение с поршнем. Пусть теперь $\tau_{bottom} > 0$, то есть частица летит вниз. Тогда все зависит от того, что произойдет раньше: частица долетит до дна или ее догонит поршень. Иными словами, если $\tau_{bottom} < \tau_{piston}$, то произойдет столкновение с дном, в противном случае – с поршнем.

```
if (tau_bottom > 0 && tau_bottom < tau_piston)
{
  // . . .  (обработка случая столкновения с дном)
}
else
{
  // . . .  (обработка случая столкновения с поршнем)
}
```

Удобно ввести настоящее время до столкновения `tau_collision`, которое будет приравниваться `tau_bottom` в первом случае и `tau_piston` во втором. Теперь нужно понять, что происходит, если реализуется первый случай. Во-первых, тела совершают свое движение до момента столкновения: координата частицы `x` убывает до нуля, ее скорость `v` не меняется, координата `X` и скорость `V` поршня меняются по законам равноускоренного движения. Получаем:

```
// обработка случая столкновения с дном:
tau_collision = tau_bottom;
x = 0;   // известно из того, что происходит
         // столкновение с дном; скорость частицы
         // в процессе движения не меняется
X = X + V * tau_collision
    - g * tau_collision * tau_collision / 2.;
V = V - g * tau_collision;
// равноускоренное движение поршня
// . . .
```

Далее, при самом столкновении движение поршня не подвергается изменениям, а частица отскакивает от нижней стенки со скоростью, противоположной исходной:

```
v = -v;   // отскок от бесконечно массивной стенки
```

Аналогично, в случае столкновения с поршнем движение тел до столкновения рассчитывается по известным законам:

```
// обработка случая столкновения с поршнем:
tau_collision = tau_piston;
x = x + v * tau_collision;
X = x;   // известно из того, что происходит
         // столкновение поршня и частицы
v_before = v;  V_before = V - g * tau_collision;
// в момент перед столкновением
// . . .
```

Здесь мы ввели переменные `v_before` и `V_before`, обозначающие скорости частицы и поршня непосредственно перед столкновением (v_b и V_b в последующих формулах). Далее, выведем формулы для скоростей этих тел после столкновения, используя законы сохранения импульса и энергии. Из системы

$$mv + MV = mv_b + MV_b,$$

$$\frac{mv^2}{2} + \frac{MV^2}{2} = \frac{mv_b^2}{2} + \frac{MV_b^2}{2}$$

находим:

$$v = \frac{(m-M)v_b + 2MV_b}{M+m},$$

$$V = \frac{(M-m)V_b + 2mv_b}{M+m}.$$

На основе этого пишем пересчет скоростей в результате столкновения, завершающий обработку случая столкновения с поршнем:

```
v = ( (m - M) * v_before + 2 * M * V_before )
    / (M + m);
V = ( (M - m) * V_before + 2 * m * v_before )
    / (M + m);
// после столкновения
```

Остается только в обоих случаях обновить значение полного времени движения, прибавив к нему величину обработанного участка:

```
t = t + tau_collision;
```

и функция `Evolution_until_after_next_collision` реализована.

Итак, мы имеем основной цикл вычислений, в котором многократно повторяется применение функции, реализующей переход от столкновения к столкновению. Таким образом, мы создали компьютерную модель изучаемого явления, способную развиваться во времени, имитируя движение нашей механической системы. То есть, мы реализовали первую часть нашего плана построения моделирующей программы. Далее, нам необходимо осуществить вторую часть, а именно создать средства, позволяющие нам посмотреть на поведение компьютерной модели и увидеть некоторые закономерности, которые помогут нам лучше понять процессы, происходящие в реальной[1)] механической системе. Более того, при создании этих средств мы часто собственно выбираем, за какими именно характеристиками системы мы будем следить, то есть более точно определяем те аспекты явления, которые мы собираемся изучать. Ведь характерная черта исследовательской работы как раз и состоит в том, что заранее неизвестно, какого именно рода закономерности (или их отсутствие) откроются при работе над задачей.

Ввиду сказанного, будем вводить механизмы наблюдения над системой постепенно. Для начала можно просто посмотреть за изменением одного из параметров во времени. Например, мы можем проследить за эволюцией высоты `X` поршня, играющей роль объема газа. Ясно, что легче всего получить ряд значений этой величины в моменты после столкновений. С этого и имеет смысл начать, имея в виду разумное предположение, что столкновения будут происходить с большой частотой, и мы получим достаточно подробную выборку.

В какой форме следует выводить эти значения? Мы собираемся отображать одну величину (высоту поршня) как функцию другой (времени). Для визуализации функциональных зависимостей одним из наиболее наглядных способов в большинстве случаев является изображение графика функции. По графику,

[1)] Мы называем здесь механическую систему реальной с точки зрения компьютерного моделирования, хотя, при противопоставлении моделей вообще явлениям окружающего мира, эта система, конечно же, сама является моделью.

как правило, нетрудно определять простые общие характеристики функции: изменения знака, возрастание – убывание, выпуклость, периодичность и т.п. Далее, для анализа более тонких свойств могут применяются другие методы.

Итак, начнем с вывода функции $X(t)$ в виде графика. Для этого есть, по крайней мере, две возможности. Первый способ – выводить точки графика в процессе вычисления. Его обычно имеет смысл применять в тех случаях, когда достаточно большая часть времени работы программы происходит под наблюдением человека (то есть для не слишком длительных вычислений). Более того, если предполагается возможность управления программой в процессе вычислений, то фактически этот способ становится единственно возможным. Второй способ состоит в том, что во время выполнения программы значения выводятся в файл (или сохраняются в памяти, а в конце выводятся в файл – чтобы минимизировать количество обращений к диску, которые занимают существенное количество времени), а рисование графиков происходит уже после завершения счета. Минусом его является потеря интерактивности. Однако он имеет и свои плюсы: вычисление и анализ его результатов могут быть разнесены во времени. Это имеет смысл для длительных вычислений, а также для того, чтобы вести архив результатов расчетов и впоследствии иметь возможность различными способами их отображать. Кроме того, при этом способе вычисление и изображение могут осуществляться в разных программах, и поэтому изображающую программу необязательно создавать, а можно воспользоваться уже готовыми программными средствами. Это имеет существенные преимущества, так как реализация средств графического отображения высокого качества – весьма нетривиальная задача, а готовые средства такого рода существуют (например, графики в данной книге получены с помощью программного пакета Mathematica; в последней главе объяснено, как это осуществить; для этой же цели применимы также программы gnuplot, Microsoft Excel и многие другие).

Ввиду этих обстоятельств мы в данной книге будем пользоваться вторым способом, то есть выводить данные, предназначенные для графического отображения, в файл, а затем отдельно на их основе рисовать графики. Для этого в функции `main()`

до основного цикла создадим файловые потоки для вывода:

```
ofstream X_res_file("X.res");
ofstream t_res_file("t.res");
```

сразу же напишем в конце функции `main()` операторы, их закрывающие:

```
X_res_file.close();
t_res_file.close();
```

и добавим в основном цикле оператор вывода значений величин `X` и `t` в файлы:

```
  X_res_file << X << ", ";
  t_res_file << t << ", ";
```

Теперь, когда мы имеем компьютерную модель нашей системы и средство наблюдения за ней, мы можем осуществить первый осмысленный запуск программы. Для этого нам необходимо, во-первых, задать значения параметров задачи, а именно массы частицы и поршня и ускорение свободного падения поршня. Для простоты сделаем это с помощью определения констант (в заголовке программы):

```
#define m 1
#define M 1000
#define g 1.
```

Здесь мы хотим отразить тот факт, что масса частицы много меньше массы поршня. При этом важны только соотношения величин одинаковой размерности. Абсолютные значения параметров не несут в себе важного смысла, так как они определяются выбором единиц измерения. Во-вторых, мы должны перед началом вычислений задать начальные значения динамических переменных, то есть начальные высоту и скорость частицы и поршня. Это легче всего[1)] сделать простым присваиванием значений в начале функции `main`:

```
t = 0.;
x = 1.; v = 100.;
X = 20.; V = 0.;
```

[1)] В более крупных задачах предпочтительней считывать значения параметров и начальные значения переменных из конфигурационного файла, что позволяет изменять их без необходимости изменения и перекомпиляции программы.

Мы задали начальную скорость поршня равной нулю, имея в виду то, что нас интересует его движение, обусловленное ударами частицы, а не собственно движение в поле тяжести. Тем не менее это решение отнюдь не является обязательным. Выбор значений остальных параметров на данном этапе также в большой степени произволен.

Таким образом, наша программа приобрела следующий вид:

```
#include <math.h>
#include <fstream>

using namespace std;

#define m 1
#define M 1000
#define g 1.

// Функция, пересчитывающая координаты частицы и поршня
// к моменту сразу после следующего столкновения
void evolution_until_after_next_collision(double& x,
          double& v, double& X, double& V, double& t)
{
  double tau_bottom = - x / v;
  // время до возможного столкновения с дном

  double D = (v-V) * (v-V) - 2. * g * (x - X);
  double tau_piston = (V - v + sqrtl(D)) / g;
  // время до возможного столкновения с поршнем

  double tau_event;
  // время до ближайшего столкновения

  double v_before, V_before;
  // скорости в момент перед столкновением с поршнем

  if (tau_bottom > 0 && tau_bottom < tau_piston)
  {  // случай столкновения с дном
    tau_event = tau_bottom;
```

```
    x = 0;   // известно из того, что
             // происходит столкновение с дном
    v = -v;   // частица отскакивает от
             // бесконечно массивной стенки
    X = X + V * tau_event
        - g * tau_event * tau_event / 2.;
    // равноускоренное движение
    V = V - g * tau_event;
  }
  else   // случай столкновения с поршнем
  {
    tau_event = tau_piston;
    x = x + v * tau_event;
    X = x;   // известно из того, что происходит
             // столкновение поршня и частицы
    v_before = v;  V_before = V - g * tau_event;
    // в момент перед столкновением
    v = ( (m - M) * v_before + 2 * M * V_before )
        / (M + m);
    V = ( 2 * m * v_before + (M - m) * V_before )
        / (M + m);
    // после столкновения
  }

  t = t + tau_event;
  // полное время движения увеличивается
  // на длину обсчитанного промежутка
}


int main()
{
  double t;
  // время (отсчитывается с начала расчета)
  double x, v;   // координата и скорость частицы
  double X, V;   // координата и скорость поршня

  t = 0.;
```

```
  x = 1.; v = 100.;
  X = 10.; V = 0.;

  ofstream X_res_file("X.res");
  ofstream t_res_file("t.res");

// рассчитываем движение на промежутке,
// содержащем 500 столкновений
  for (int n = 1; n <= 500; n++)
  {
    evolution_until_after_next_collision(x,v,X,V,t);

  // обработка полученных данных
    X_res_file << X << ", ";
    t_res_file << t << ", ";
  }

  X_res_file.close();
  t_res_file.close();
  return 0;
}
```

1.4.3. Исследование задачи с помощью программы

Итак, осуществим запуск программы. Мы получим в файле `X.res` следующий ряд значений переменной `X`:

`19.9820, 19.9642, 19.9068, 19.8504, 19.7558, . . .`

а в файле `t.res` – значения переменной `t`:

`0.18982, 0.389282, 0.587994, 0.785572, 0.981651, . . .`

Теперь построим по этим точкам график функции $X(t)$. Это можно сделать с помощью одной из стандартных программ, поддерживающих построение графика по набору пар координат точек. Получим результат, показанный на рис. 1.1. Напомним, что точки на нем соответствуют моментам столкновений.

Видно, что сначала поршень начинает опускаться. Далее в определенный момент его координата достигает минимума и затем начинает возрастать, возвращаясь к первоначальному значению. После этого повторяется аналогичный процесс, по край-

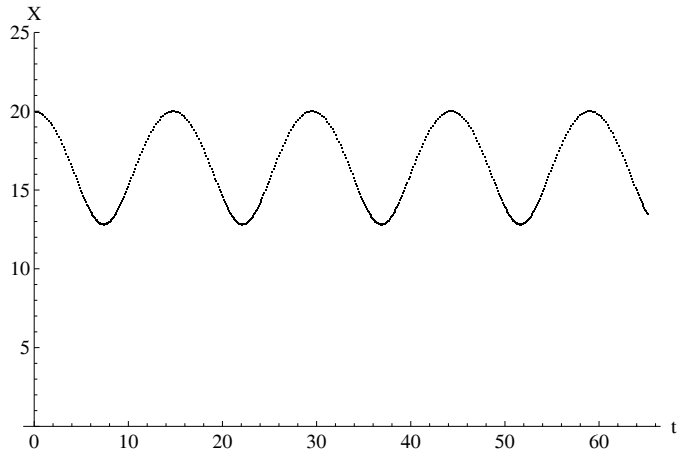


Рис. 1.1. Эволюция высоты поршня $X(t)$ при начальном коэффициенте неравновесности 2.

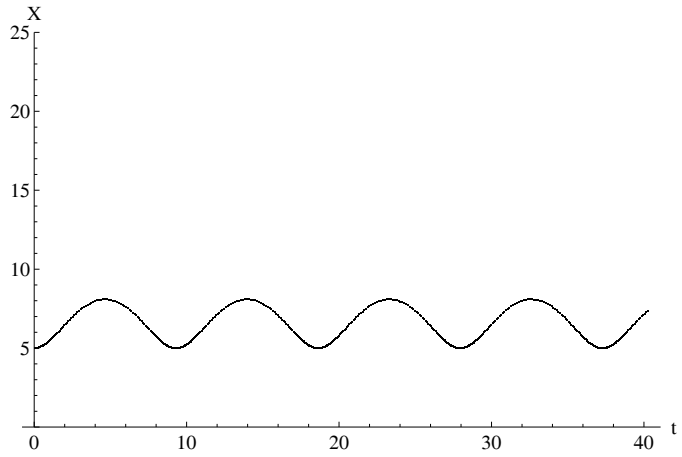


Рис. 1.2. Эволюция высоты поршня $X(t)$ при начальном коэффициенте неравновесности 0,5.

ней мере, насколько можно видеть из графика. Ясно, что в точности функция, по которой X зависит от t, может не быть периодической (как мы помним, она состоит их отрезков парабол, соединенных в точках, где происходят столкновения поршня и частицы). Однако с точки зрения крупномасштабной динамики координаты поршня можно говорить о том, что она совершает колебания с периодом около 15 единиц времени.

Если поэкспериментировать с начальными условиями, то можно обнаружить, что помимо случаев, сходных с изображенным на рис. 1.1, когда в начальный момент поршень опускается, существуют также случаи, когда динамика поршня начинается с подъема. Например, это происходит при задании начальной высоты поршня, равной 5, а остальных параметров – тех же, что и в предыдущем расчете. График для этого случая приведен на рис. 1.2. При этом тенденция, которую можно проследить, состоит в том, что динамика первого типа, то есть начальное падение поршня, имеет место, когда его начальная высота достаточно велика (при прочих неизменных параметрах), а динамика второго типа – когда высота достаточно мала. Эти факты можно сопоставить со следующим наглядным представлением о характере этой динамики, которое можно попытаться получить, анализируя нашу систему. Именно, движение поршня подвержено влиянию двух конкурирующих факторов: силе тяжести, тянущей его вниз, и ударам частицы, выталкивающим его наверх. При этом первая сила растет с увеличением массы поршня M и ускорения его свободного падения g, а вторая возрастает с увеличением массы частицы m и ее скорости v и уменьшается с ростом высоты поршня X, так как при этом полет частицы между двумя ударами о поршень занимает больше времени, а значит удары происходят реже. В связи с этим можно предположить, что если в начальный момент первый фактор пересиливает второй, то есть слишком велики M, g и X или же слишком малы m и v, то поршень начинает опускаться, а в обратном случае – подниматься. Кроме того, можно предположить, что существует промежуточный случай, когда оба фактора скомпенсированы, и тогда гипотеза состоит в том, что высота поршня не будет существенно меняться.

При каких именно условиях это должно происходить? Здесь-то и возникает идея попытаться обратиться к условию, выведенному для равновесного состояния газа. В соответствии с уравнением (1.5) имеем:

$$P = \frac{2}{V}\,\frac{m\,v^2}{2}.$$

Коэффициент 3 в нашем случае не возникает, так как движение одномерно и скорость имеет одну компоненту. Отсюда, сопоставляя давлению силу, действующую на поршень, а объему – его высоту, получаем условие:

$$MgX = mv^2, \tag{1.10}$$

которое мы хотим протестировать как возможный критерий того, что поршень в процессе динамики не совершает ни сильного падения, ни сильного подъема. Задавая, например, те же параметры, что и раньше, но с высотой поршня, равной 10 (что дает соответствие с (1.10)), получаем динамику, показанную на рис. 1.3. Видим, что, как и ожидалось, высота поршня существенно не меняется.

1.4.4. Выводы

Таким образом, мы ответили на вопрос, поставленный в начале обсуждения этой задачи: случай, подчиненный условию – аналогу уравнения состояния идеального газа, действительно обладает в нашей задаче особыми свойствами: это условия, при которых действие ударов частицы на поршень компенсирует действие постоянной силы, и в итоге высота поршня остается вблизи своего начального значения. Следует, однако, заметить, что напрямую применять утверждения об идеальном газе к нашей системе некорректно, так как в случае газа мы имели дело с большим ансамблем частиц, и параметры этой системы получались как статистические средние величин в этом ансамбле. Именно в таком понимании можно говорить о единой температуре для всего объема газа и о едином давлении во всех точках его поверхности. В нашем же случае речь идет об одной частице, поэтому полученный результат следует рассматривать скорее как аналогию, а не как прямое применение выводов, полученных для идеального газа.

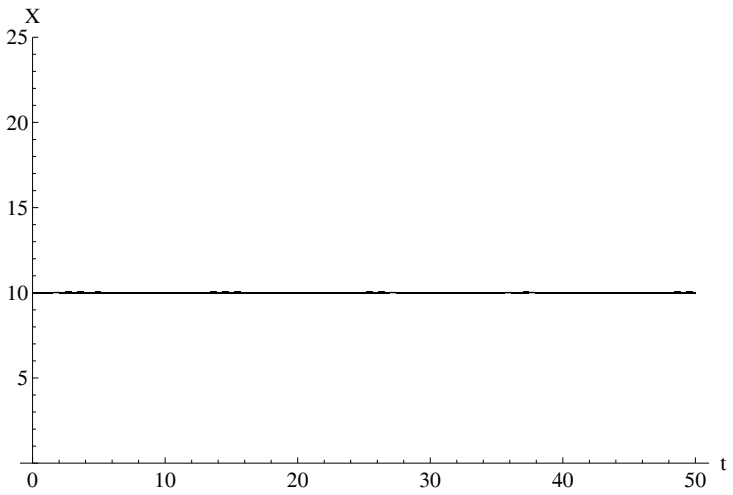


Рис. 1.3. Эволюция высоты поршня $X(t)$ в равновесии (при начальном коэффициенте неравновесности 1): поршень колеблется с очень маленькой амплитудой.

Глава 2

Резонанс

Явление резонанса играет очень большую роль в динамике, его исследованию посвящено большое число работ. Здесь мы ограничимся разбором наиболее простой ситуации резонанса для одномерного гармонического осциллятора.

2.1. Силовой резонанс

Рассмотрим одномерное движение материальной точки под действием упругой силы, подчиняющейся закону Гука, и внешней силы $f(t)$, зависящей от времени, при наличии диссипации. Последнюю будем считать пропорциональной скорости частицы, т.е. уравнения движения имеют вид

$$\ddot{x} + \gamma\,\dot{x} + \omega^2\,x = f(t) \tag{2.1}$$

Обозначим через $x_1,\ x_2$ фундаментальные решения однородной системы, когда $f = 0$, так что общее решение имеет вид

$$x(t) = C_1\,x_1(t) + C_2\,x_2(t)$$

Для решения уравнения (2.1) применим метод вариации постоянных, т.е. предположим, что константы $C_1,\ C_2$ *зависят от времени*. Для того, чтобы их можно было найти, необходимы дополнительные условия, которые мы выберем в виде

$$\frac{dC_1}{dt}\,x_1 + \frac{dC_2}{dt}\,x_2 = 0 \tag{2.2}$$

Благодаря (2.2) выражение для первой производной имеет вид

$$\frac{dx}{dt} = C_1\,\frac{dx_1}{dt} + C_2\,\frac{dx_2}{dt}$$

и соответственно для второй производной

$$\frac{d^2x}{dt^2} = C_1\,\frac{d^2x_1}{dt^2} + C_2\,\frac{d^2x_2}{dt^2} + \frac{dx_1}{dt}\,\frac{dC_1}{dt} + \frac{dx_2}{dt}\,\frac{dC_2}{dt}$$

Подставляя указанные выражения производных в уравнение осциллятора, получаем ещё одно уравнение для $C_1,\ C_2$

$$\frac{dC_1}{dt}\,\frac{dx_1}{dt} + \frac{dC_2}{dt}\,\frac{dx_2}{dt} = f(t) \tag{2.3}$$

Из уравнений (2.2)(2.3) получаем

$$\frac{dC_1}{dt} = -\frac{x_2\, f(t)}{x_1\,\dot{x}_2 - x_2\,\dot{x}_1}, \quad \frac{dC_2}{dt} = \frac{x_1\, f(t)}{x_1\,\dot{x}_2 - x_2\,\dot{x}_1},$$

откуда имеем частное решение уравнения осциллятора

$$x(t) = -x_1 \int \frac{x_2\, f(t)}{x_1\,\dot{x}_2 - x_2\,\dot{x}_1}\, dt + x_2 \int \frac{x_1\, f(t)}{x_1\,\dot{x}_2 - x_2\,\dot{x}_1}\, dt \tag{2.4}$$

Для того, чтобы понять характер решения, воспользуемся тем, что

$$x_{1,2} = e^{\pm\lambda_{1,2}\, t}, \qquad \lambda_{1,2} = -\frac{\gamma}{2} \pm i\,\sqrt{\omega_0^2 - \frac{\gamma^2}{4}}$$

Решение (2.4) примет вид

$$x = \frac{e^{-\dfrac{\gamma t}{2}}}{2i\,\Omega}\left[e^{i\Omega\, t}\ \int e^{\dfrac{\gamma t}{2}}\ e^{i\Omega t}\, f(t)\, dt + H.C.\right] \tag{2.5}$$

где

$$\Omega = \sqrt{\omega_0^2 - \frac{\gamma^2}{4}}$$

В случае, когда диссипация отсутствует ($\gamma = 0$), а внешняя сила $f(t)$ содержит составляющую $\propto \cos(\omega_0\, t)$, возникает резонанс, что приводит к неограниченному возрастанию амплитуды колебаний (подробное описание см. в следующем разделе). Диссипация за счёт затухающего экспоненциального множителя $\exp(-\gamma\, t/2)$ не даёт этому процессу развиться и обеспечивает конечность амплитуды колебаний. Отметим, что приведённый вывод не требует изначального предположения касательно синусоидального характера возбуждающей силы, которая может, в частности, иметь характер отдельных щелчков, с периодом, совпадающим с периодом свободных колебаний осциллятора.

2.2. Задача для компьютерного моделирования: силовой резонанс для осциллятора с диссипацией

2.2.1. Мотивировки и постановка задачи

Одномерный гармонический осциллятор является модельной системой, на которой можно наблюдать основные явления, связанные с колебаниями. При этом, вообще говоря, колебательные явления могут иметь весьма различную природу: колебания параметров орбит планет, звуковые явления, осцилляции токов в электрических цепях, колебания численности популяций животных и т. д. Кроме того, известно, что во многих реальных ситуациях существенное влияние на динамику оказывает дисспация – утечка энергии из колебательного движения в другие ее формы. Поэтому целесообразно рассмотреть в качестве модельной системы осциллятор с диссипацией (сохраняя, разумеется, возможность задать ее малой или даже нулевой). Конкретный вид диссипации, вообще говоря, существенно зависит от реальной ситуации и может даже потребовать одновременного расчета динамики той системы (или группы степеней свободы той же системы), в которую уходит энергия. Однако в довольно распространенном случае, когда диссипацию можно свести к действию на колебательную систему силы трения, зависящей только от скорости, $F_{diss}(\dot{x})$, и когда скорость можно считать малой, – в этом случае силу трения можно разложить по степеням $\dot{x}$ и оставить главную часть, которая оказывается линейной: $F'_{diss}(0)\,\dot{x}$ (так как в покое трения нет, а значит, $F_{diss}(0) = 0$). Именно таким трением, пропорциональным скорости, мы и ограничимся в данной задаче.

Движение предложенной системы описывается уравнением (2.1):

$$\ddot{x} + \gamma\,\dot{x} + \omega_0^2\,x = f(t).$$

Его решение, полученное в предыдущем пункте, задается формулой (2.5). Однако отнюдь не во всех задачах удается прийти к подобному результату: методы поиска решений дифференциальных уравнений в виде формул далеко не всегда приводят к успеху. Более того, современная наука приходит к выводу,

что задачи, где это можно сделать, составляют в определенном смысле исключение, а не правило. Кроме того, даже если ответ получен в виде формул, но они имеют очень сложный вид, то анализ свойств решения все равно может быть сильно затруднен (например, если в рассматриваемой задаче интеграл не берется в элементарных функциях). Ввиду этого оказывается весьма полезным применить численное решение дифференциальных уравнений задачи, позволяющее получить последовательность значений динамических переменных (для дискретных моментов времени), которая близка к соответствующей выборке из точного решения. На основе этой последовательности можно делать определенные выводы о характере динамики, а также, при благоприятной ситуации с точностью численного расчета, производить количественные предсказания.

В данной задаче, однако, аналитические методы все же позволяют получить решение (в разумном виде) в определенном классе случаев, важном для многих сфер применения этой теории. Поэтому у нас будет возможность сравнить результаты, даваемые численными методами, с точными решениями.

Итак, перейдем к созданию компьютерной модели изучаемой системы. Для моделирования ее динамики нам потребуется применить численный метод решения задающего ее дифференциального уравнения. Для этого его удобно привести к виду системы уравнений первого порядка:

$$\begin{aligned} \dot{x} &= v, \\ \dot{v} &= -\gamma\, v - \omega_0^2\, x + f(t), \end{aligned} \tag{2.6}$$

так как методы численного интегрирования приводятся обычно для таких систем. Одним из наиболее простых и вместе с тем достаточно надежных алгоритмов является так называемый метод трапеций.

2.2.2. Метод трапеций решения систем обыкновенных дифференциальных уравнений

Рассматривается система уравнений первого порядка:

$$\dot{\vec{x}} = \vec{F}(\vec{x}, t),$$

где $\vec{x} = (x_1, \ldots, x_k)$ – вектор искомых функций (динамических переменных в задачах механики) от независимой переменной t (времени). Задаем последовательность моментов времени:

$$t_n = n\,\Delta t,$$

в которые будут вычисляться значения динамических переменных; здесь Δt – так называемый шаг интегрирования. Ясно, что он должен быть, по крайней мере, меньше, чем характерный временной масштаб динамики системы. Будем обозначать вектор значений динамических переменных $\vec{x}$, вычисленных для момента времени t_n, как $\vec{x}_n$ (не путать с обозначением компонент, где используется буква без стрелки). Алгоритм численного интегрирования позволяет по приближенно вычисленным ранее значениям переменных $\vec{x}_n$ на данный момент вычислить приближенные значения переменных $\vec{x}_{n+1}$ в следующий момент времени. При этом используются алгебраические операции и одно или несколько применений функции $\vec{F}(\vec{x}, t)$. Кроме того, в так называемых многошажных методах могут использоваться значения динамических переменных для нескольких значений времени, предшествующих данному. Критерием качества работы метода является близость вычисленных значений $\vec{x}_n$ к значениям $\vec{x}(t_n)$ точного решения задачи в соответствующие моменты времени.

Метод трапеций строится на основе приближенной формулы:

$$\vec{x}_{n+1} - \vec{x}_n = \frac{1}{2}\left[\vec{F}(\vec{x}_n, t_n) + \vec{F}(\vec{x}_{n+1}, t_{n+1})\right]\,\Delta t. \qquad (2.7)$$

Это соотношение, в случае, если $\vec{F}(\vec{x}, t)$ не зависит от $\vec{x}$ и все векторы одномерны, соответствует формуле трапеций вычисления определенного интеграла от функции $F(t)$, которая основана на приближенном вычислении площади криволинейной трапеции под графиком функции с основаниями $F(t_n)$ и $F(t_{n+1})$ и высотой Δt по формуле площади обыкновенной трапеции.

Особенность уравнения (2.7) состоит в том, что величина $\vec{x}_{n+1}$, которую мы собираемся вычислить (по известному на данный момент $\vec{x}_n$), входит в него не только в левой части, но и в правой как аргумент функции $\vec{F}$. Методы такого типа называются неявными. Оказывается, что именно они способны сохра-

нять точность вычислений для систем, представляющих трудности при численном интегрировании (так называемые жесткие системы).

Итак, необходимо найти значение $\vec{x}_{n+1}$ из уравнения (2.7). Для этого обычно применяют метод простых итераций. Именно, задаются некоторым стартовым значением $\vec{x}^{(0)}_{n+1}$ (например, можно использовать прошлое значение $\vec{x}_n$), а затем последовательно находят значения $\vec{x}^{(1)}_{n+1}, \vec{x}^{(2)}_{n+1}, \ldots$, все ближе подходящие к искомой величине $\vec{x}_{n+1}$, по следующему правилу:

$$\vec{x}^{(i+1)}_{n+1} = \vec{x}_n + \frac{1}{2}\left[\vec{F}(\vec{x}_n, t_n) + \vec{F}(\vec{x}^{(i)}_{n+1}, t_{n+1})\right]\,\Delta t.$$

Остановка процесса производится в тот момент, когда приращение $\vec{x}^{(i+1)}_{n+1} - \vec{x}^{(i)}_{n+1}$ становится мало в некоторой норме (например, по евклидовой длине). Мы не имеем возможности подробно обсуждать, для какого класса случаев этот процесс действительно сходится к искомому значению, и с какой скоростью. Для практических целей, однако, можно отметить, что в большинстве случаев, для нежестких систем, уже несколько итераций приводят к невязкам, сравнимым с машинными ошибками округления.

Таким образом, с помощью описанной процедуры мы можем вычислять значения динамических переменных $\vec{x}_{n+1}$ в следующий момент времени. Повторное применение этой процедуры N раз дает дискретную траекторию с шагом Δt на интервале времени $(0, N\Delta t)$. Теоретическое определение того, насколько она будет близка к точному решению, – вопрос довольно непростой и опять же выходящий за рамки этой книги. Однако на конкретных примерах мы будем иметь возможность наблюдать, каковы порядки близости численных решений к точным и как они зависят от параметров системы и величины шага интегрирования.

Теперь мы можем приступить к составлению программы, численно решающей уравнение (2.1).

2.2.3. Программная реализация.

Структура программы, решающей данную задачу, является довольно типичной. Именно, блок, осуществляющий численное

интегрирование дифференциальных уравнений, имеет похожий вид для большинства задач, в которых он встречается. Этот блок оформляется в виде основного цикла вычислений, в каждом повторении которого осуществляется один временной шаг алгоритма интегрирования[1)]:

```
while (t < 100)   // рассчитываем траекторию
{    // на интервале времени (0, 100)
    // ... (шаг интегрирования)
}
```

Шаг интегрирования изменяет значения переменных (`x` и `v` в нашем случае) в соответствии с используемым численным алгоритмом. Так же как в формулах удобно все переменные решаемой системы дифференциальных уравнений записывать в виде вектора $\vec{x}$, так и в программе весьма желательно собрать соответствующие переменные в единое целое. Программным аналогом формульных обозначений вида $x_1, \ldots, x_k$ являются массивы: `x[0],..., x[k-1]`. Однако если (как, например, в нашей задаче) исходно переменные являлись разнородными, назывались разными буквами, то полезно сохранить эти названия и в программе. Для этого можно использовать механизм структур (или классов). Они обладают также тем важным преимуществом, что для них можно определить действие операторов, в частности, арифметических действий, что делает написание и восприятие алгебраических выражений с их участием гораздо нагляднее и проще.

Итак, определим структуру, содержащую динамические переменные нашей задачи:

```
struct Var   // переменные системы первого порядка
{
    double x, v;
};
```

В функции `main()`, до основного цикла, объявим переменную этого типа:

```
Var V;
```

(не путать имя структуры `V` с ее полем `v`, содержащим скорость частицы). В алгоритме интегрирования используются арифме-

[1)] Полный текст программы см. на стр. 52.

тические операции сложения двух векторов переменных и умножения вектора переменных на число. Используя механизм перегрузки операторов, мы можем реализовать эти операции для нашей структуры следующим образом:

```
Var operator +(Var& A, Var& B)
{   // сложение двух структур переменных
    Var Res;
    Res.x = A.x + B.x;
    Res.v = A.v + B.v;
    return Res;
}

Var operator *(double k, Var& A)
{   // умножение структуры переменных на число
    Var Res;
    Res.x = k * A.x;
    Res.v = k * A.v;
    return Res;
}
```

Здесь и далее мы передаем структуры по ссылке (даже если функция не должна изменять их), так как это повышает временну́ю эффективность (не тратится время на создание и заполнение локальной переменной). Для структур большого размера этот эффект может быть существенным.

Следующее, что нам необходимо для реализации шага алгоритма интегрирования, – это функция, соответствующая $\vec{F}(\vec{x}, t)$, которая задает первые производные по времени от переменных задачи, то есть, в нашем случае, правая часть уравнений (2.6), задающих нашу задачу в виде системы первого порядка. В программе назовем эту функцию `RHS(...)` (right-hand side, правая часть). В качестве аргументов в эту функцию нужно передать, во-первых, посчитанную на данный момент структуру `V` и текущее время `t`. Во-вторых, необходимо передать значения параметров нашей задачи: ω_0 и γ. Для этого удобно их также собрать в структуру, учитывая, кроме того, что при определении выражения для внешней силы в нем также может потребоваться использовать некоторые дополнительные константы, которые мы сможем просто добавить в структуру,

не меняя уже написанный использующий ее код.

Итак, на данном этапе определяем структуру параметров вида:

```
struct Par   // структура параметров задачи
{
    double omega0, gamma;
};
```

Теперь мы можем написать функцию `RHS`. В соответствии с нашей системой (2.6) имеем:

```
Var RHS(Var& V, Par& P, double t)
{   // правые части системы первого порядка
    Var Res;
    Res.x = V.v;
    Res.v = - P.gamma * V.v - pow(P.omega0, 2) * V.x
            + Ext_force(P, t);
    return Res;
}
```

Здесь `Ext_force(P, t)` – функция, задающая внешнюю силу $f(t)$. В нее передается структура `P`, в которую мы позднее включим параметры, калибрующие выражение для внешней силы. Пока можно задать ее произвольной формулой без параметров.

Теперь напишем функцию, реализующую шаг метода трапеций:

```
void Trapezium_step(Var& V, Par& P, double& t,
                                            double st)
{
    Var V0 = V;   // структура переменных
                  // на предыдущем шаге
    Var F0 = RHS(V0, P, t);   // значение правых частей
                              // на предыдущем шаге
    Var V_prev = V;
    // вспомогательная структура, хранящая значение
    // вычисляемой новой структуры переменных,
    // полученное на предыдущей итерации

    do   // цикл решения алгебраического уравнения
    {     // методом простых итераций
```

```
        V_prev = V;

        V = V0 + 0.5 * st * ( F0
             + RHS(V_prev, P, t + st) );
    }
    while ( fabsl(V.x - V_prev.x)
          + fabsl(V.v - V_prev.v) > 0.00000001 );
  // сравнение результатов текущей и предыдущей
  // итераций в норме, равной сумме модулей компонент;
  // вычисление останавливается, если норма разности
  // меньше заданного значения

    t += st;   // сдвиг текущего времени на шаг
}
```

Здесь мы запоминаем значение структуры переменных на предыдущем шаге в структуру `V0`, а затем в цикле осуществляем решение уравнения метода трапеций с помощью итераций. Чтобы иметь возможность остановить вычисление, когда очередная итерация не приводит к существенному изменению результата, мы вводим переменную `V_prev`, хранящую результат предыдущей итерации. Результатом работы этой функции является то, что структура `V` содержит новые значения переменных, соответствующие времени, увеличенному на шаг интегрирования по сравнению с предыдущим значением.

Теперь мы можем написать действие, которое должно выполняться в основном цикле вычислений: это реализованный нами шаг интегрирования. Итак, получаем основной цикл:

```
    while (t < 100)   // рассчитываем траекторию
    {                 // на интервале времени (0, 100)
        Trapezium_step(V, P, t, st);
        // шаг интегрирования
    }
```

(здесь `P` – структура параметров, `st` – величина шага интегрирования).

Итак, мы реализовали первую задачу моделирующей программы, состоящую в создании в памяти компьютера модели изучаемого явления, развивающейся в соответствии с его законами. Приступим ко второй задаче, а именно, к созданию

средств наблюдения за этой моделью, позволяющих выявлять закономерности ее поведения.

Как и в предыдущей задаче, простейшим из таких средств является вывод значений координаты `x` частицы после каждого шага или с некоторым более длительным интервалом, если нет необходимости в большой детальности получаемой траектории. Например, будем выводить координату каждые 100 шагов (что дает достаточную детализацию при условиях задачи, описываемых далее). Для этого введем счетчик шагов `i`, который при достижении значения 100 будем обнулять и выводить координату частицы в файл:

```
// в начало основного цикла:
i++;   // счетчик шагов

if ( i == 100 )
{   // каждые 100 шагов выводим координату в файл
    x_res_file << V.x << " ";
    i = 0;
}
```

Теперь можно переходить к расчетам. Для этого, прежде всего, надо определиться с видом внешней силы, действующей на осциллятор. Здесь мы хотим рассмотреть важный случай, когда внешняя сила сама имеет характер колебания:

$$f(t) = F\cos\omega t.$$

Такой вид воздействия часто встречается в конкретных физических колебательных системах. Кроме того, этот случай удобен для нас тем, что в нем задача нахождения динамики полностью решается аналитически, что даст нам возможность сравнивать численное решение с полученным с помощью формул. Для нахождения частного решения неоднородного уравнения в этом случае[1)] можно применить более простой метод[2)], чем изложенный в предыдущем пункте метод вариации постоянных.

[1)] А также в более общей ситуации, когда правая часть имеет вид так называемого обобщенного полинома: $P_k(t)\,e^{\lambda t}$, где $P_k(t)$ – полином некоторой степени k, λ – комплексное число.

[2)] Называемый обычно методом мод, или методом неопределенных коэффициентов.

А именно, будем искать частное решение в виде:

$$x_{part}(t) = p\cos\omega t + q\sin\omega t,$$

где p и q – неопределенные коэффициенты. Подставляя в уравнение движения (2.1) и приравнивая коэффициенты при $\cos\omega t$ и $\sin\omega t$, получаем как решение линейной алгебраической системы:

$$p = \frac{F\left(\omega_0^2 - \omega^2\right)}{\left(\omega_0^2 - \omega^2\right)^2 + \gamma^2\omega^2}, \quad q = \frac{F\gamma\omega}{\left(\omega_0^2 - \omega^2\right)^2 + \gamma^2\omega^2}.$$

Таким образом, получаем общее решение:

$$x_{gen}(t) = e^{-\gamma t/2}(a\cos\Omega t + b\sin\Omega t) + \frac{F\left[\left(\omega_0^2 - \omega^2\right)\cos\omega t + \gamma\omega\sin\omega t\right]}{\left(\omega_0^2 - \omega^2\right)^2 + \gamma^2\omega^2},$$

где, как и раньше, $\Omega = \sqrt{\omega_0^2 - \frac{\gamma^2}{4}}$, a и b – постоянные, определяемые из начальных условий.

Для нахождения частного решения с заданными начальными условиями $x(0) = x_0$ и $\dot{x}(0) = \dot{x}_0$ удобно сначала найти частное решение с нулевыми начальными условиями: $x(0) = 0$ и $\dot{x}(0) = 0$ (решая получающееся уравнение на a и b):

$$\begin{aligned} x_{part,0}(t) &= \frac{F}{\left(\omega_0^2 - \omega^2\right)^2 + \gamma^2\omega^2} \times \\ &\times \left[\left(\omega_0^2 - \omega^2\right)\left(\cos\omega t - e^{-\gamma t/2}\cos\Omega t\right) + \right. \\ &\left. + \gamma\omega\left(\sin\omega t - \frac{\omega_0^2 + \omega^2}{2\omega\Omega}e^{-\gamma t/2}\sin\Omega t\right)\right], \end{aligned} \tag{2.8}$$

а затем найти частное решение однородного уравнения при данных начальных условиях:

$$x_{hom}(t) = e^{-\gamma t/2}(x_0\cos\Omega t + \frac{\dot{x}_0 + \gamma x_0/2}{\Omega}\cos\Omega t). \tag{2.9}$$

Тогда искомое частное решение неоднородного уравнения запишется в виде:

$$x_{part}(t) = x_{part,0}(t) + x_{hom}(t). \tag{2.10}$$

Таким образом, в нашей программе пишем функцию, задающую внешнюю силу, в виде:

```
double Ext_force(Par& P, double t)
{  // внешняя сила
    return P.F * cosl(P.omega * t);
}
```

Здесь `P.F` и `P.omega` – параметры (соответствующие F и ω в формулах), которые надо добавить в структуру параметров `P`. Кроме того, для сравнения траекторий[1)] частицы, полученных численным интегрированием и по приведенным аналитическим формулам, напишем также функцию `Exact_x`, вычисляющую аналитическое решение в соответствии с (2.10):

```
double Exact_x(Var& V0, Par& P, double t)
{   // частное решение неоднородного уравнения
    // с заданными начальными условиями
    return Homogeneous_x(V0, P, t) + Partial_0_x(P, t);
}
```

с использованием вспомогательных функций `Homogeneous_x` и `Partial_0_x`, вычисляемых в соответствии с (2.9) и (2.8):

```
double Homogeneous_x(Var& V0, Par& P, double t)
{   // частное решение однородного уравнения
    // с заданными начальными условиями
    double Omega = sqrtl( pow(P.omega0, 2)
                   - pow(P.gamma, 2) / 4. );
    return expl(- P.gamma * t / 2.)
            * ( V0.x * cosl(Omega * t)
            + ( (V0.v + P.gamma * V0.x / 2.) / Omega )
                * sinl(Omega * t) );
}

double Partial_0_x(Par& P, double t)
{   // частное решение неоднородного уравнения
    // с нулевыми начальными условиями
    double Omega = sqrtl(  pow(P.omega0, 2)
                         - pow(P.gamma, 2) / 4. );
```

[1)] Для краткости мы называем здесь траекторией функцию $x(t)$, которая, строго говоря, носит название закона движения.

```
    double diff = pow(P.omega0,2) - pow(P.omega,2);
    double exponent = expl(- P.gamma * t / 2.);
    double factor = (pow(P.omega0,2) + pow(P.omega,2))
                    / (2. * P.omega * Omega);
    return ( P.F / (  pow(diff, 2)
             + pow(P.gamma, 2) * pow(P.omega, 2) ) )
        * ( diff * ( cosl(P.omega * t)
                   - exponent * cosl(Omega * t) )
        + P.gamma * P.omega * ( sinl(P.omega * t)
        - factor * exponent * sinl(Omega * t) ) );
}
```

Здесь, для облегчения восприятия формул, мы вычислили некоторые части выражений предварительно. С точки зрения временно́й эффективности также целесообразно вычислить те части выражений, которые не зависят от времени, один раз в начале программы и передавать в эти функции уже известные их значения. Например, можно включить соответствующие переменные в структуру `P`.

Итак, приведем полный код нашей программы. Поскольку она приобрела достаточно большой объем и, главное, стала иметь вид системы нескольких относительно независимых друг от друга блоков: алгебраические операции, интегратор, функции, задающие уравнения нашей задачи и их решения, – то целесообразно разбить ее на несколько файлов. При этом все объявления соберем в один заголовочный файл, который включим во все файлы с кодом.

Заголовочный файл `"force_res.h"`[1)]:

```
#ifndef FORCE_RES_H
#define FORCE_RES_H

struct Var   // переменные системы первого порядка
{
    double x, v;
};

// арифметические операции над структурами переменных:
```

[1)] Директивы `#ifndef` используются для предотвращения многократного включения объявлений.

```
Var operator +(Var& A, Var& B);
Var operator *(double k, Var& A);

struct Par   // структура параметров задачи
{
    double omega0, gamma, F, omega;
};

// функция, задающая внешнюю силу
double Ext_force(Par& P, double t);

// функция, вычисляющая правые части
// системы первого порядка:
Var RHS(Var& V, Par& P, double t);

// шаг метода трапеций:
void Trapezium_step(Var& V, Par& P, double& t,
                                        double st);

// функции, вычисляющие точное решение
// для используемого вида внешней силы:
double Exact_x(Var& V0, Par& P, double t);
double Homogeneous_x(Var& V0, Par& P, double t);
double Partial_0_x(Par& P, double t);

#endif
```

Файл "var.cpp", в котором определяются алгебраические операции над структурами Var:

```
#include "force_res.h"

Var operator +(Var& A, Var& B)
{   // сложение двух структур переменных
    Var Res;
    Res.x = A.x + B.x;
    Res.v = A.v + B.v;
    return Res;
}
```

```
Var operator *(double k, Var& A)
{   // умножение структуры переменных на число
    Var Res;
    Res.x = k * A.x;
    Res.v = k * A.v;
    return Res;
}
```

Файл "integ.cpp", содержащий функцию шага метода интегрирования:

```
#include "force_res.h"
#include "math.h"

void Trapezium_step(Var& V, Par& P, double& t,
                                        double st)
{
    Var V0 = V;   // структура переменных
                  // на предыдущем шаге
    Var F0 = RHS(V0, P, t);   // значение правых частей
                              // на предыдущем шаге
    Var V_prev = V;
    // вспомогательная структура, хранящая значение
    // вычисляемой новой структуры переменных,
    // полученное на предыдущей итерации

    do   // цикл решения алгебраического уравнения
    {     // методом простых итераций
        V_prev = V;

        V = V0 + 0.5 * st * ( F0
             + RHS(V_prev, P, t + st) );
    }
    while ( fabsl(V.x - V_prev.x)
          + fabsl(V.v - V_prev.v) > 0.00000001 );
  // сравнение результатов текущей и предыдущей
  // итераций в норме, равной сумме модулей компонент;
  // вычисление останавливается, если норма разности
  // меньше заданного значения
```

```
    t += st;   // сдвиг текущего времени на шаг
}
```

Файл "force_res.cpp", содержащий функции, полностью специфичные для нашей задачи: внешняя сила, правые части и точное решение:

```
#include "force_res.h"
#include <math.h>

double Ext_force(Par& P, double t)
{  // внешняя сила
    return P.F * cosl(P.omega * t);
}

Var RHS(Var& V, Par& P, double t)
{   // правые части системы первого порядка
    Var Res;
    Res.x = V.v;
    Res.v = - P.gamma * V.v
            - pow(P.omega0, 2) * V.x + Ext_force(P, t);
    return Res;
}

double Exact_x(Var& V0, Par& P, double t)
{   // частное решение неоднородного уравнения
    // с заданными начальными условиями
    return Homogeneous_x(V0, P, t) + Partial_0_x(P, t);
}

double Homogeneous_x(Var& V0, Par& P, double t)
{   // частное решение однородного уравнения
    // с заданными начальными условиями
    double Omega = sqrtl(  pow(P.omega0, 2)
                         - pow(P.gamma, 2) / 4. );
    return expl(- P.gamma * t / 2.)
        * ( V0.x * cosl(Omega * t)
        + ( (V0.v + P.gamma * V0.x / 2.) / Omega )
                                * sinl(Omega * t) );
```

```
}

double Partial_0_x(Par& P, double t)
{   // частное решение неоднородного уравнения
    // с нулевыми начальными условиями
    double Omega = sqrtl(  pow(P.omega0, 2)
                         - pow(P.gamma, 2) / 4. );
    double diff = pow(P.omega0,2) - pow(P.omega,2);
    double exponent = expl(- P.gamma * t / 2.);
    double factor = ( pow(P.omega0,2)
       + pow(P.omega,2) ) / (2. * P.omega * Omega);
    return ( P.F / ( pow(diff, 2)
           + pow(P.gamma, 2) * pow(P.omega, 2) ) )
          *(  diff * (  cosl(P.omega * t)
                      - exponent * cosl(Omega * t) )
            + P.gamma * P.omega * ( sinl(P.omega * t)
            - factor * exponent * sinl(Omega * t) ) );
}
```

Наконец, файл "main.cpp", содержащий управляющую функцию[1)]:

```
#include "force_res.h"
#include <math.h>
#include <fstream>
#include <iostream>

using namespace std;

int main()
{
    Var V;
    V.x = 0.;   V.v = 0.;
    Var V0 = V;
    Par P;
    P.omega0 = 1.;   P.gamma = 0.05;
    P.F = 1.;   P.omega = 1.2;
    double t = 0.;
```

[1)] Выбор значений параметров будет обсуждаться далее.

```
double st = 0.001;
double x_ex = 0.; // точное решение
double max_diff = 0.;
// максимальное отклонение численного решения
// от точного (на данный момент времени)

ofstream x_res_file("x.res");
// файл численной траектории

ofstream x_exact_res_file("x_exact.res");
 // файл аналитической траектории

double t1 = st;
double error_out_interval = 1.;
// вспомогательные переменные
// для вывода на экран значения ошибки
// с данным интервалом времени error_out_interval

int i = 0;
int n_steps_out = 100;
// вспомогательные переменные
// для вывода в файл значений x
// через каждые n_steps_out шагов

while (t < 200)   // рассчитываем траекторию
{    // на интервале времени (0, 200)
    Trapezium_step(V, P, t, st);
    // шаг интегрирования

    x_ex = Exact_x(V0, P, t);
    // вычислить точное решение

    i++;
    if ( i == n_steps_out )
    { // вывод численного и точного решений в файлы
        x_res_file << V.x << " ";
        x_exact_res_file << x_ex << " ";
        i = 0;
```

```
        }

        // подсчет отклонения от точного решения
        if ( fabsl(V.x - x_ex) > max_diff)
            max_diff = fabsl(V.x - x_ex);

        if (t1 >= error_out_interval)
        {   // вывод отклонения
            cout << "\ntime: " << t
                 << "   error: " << max_diff << " ";
            t1 = st;
        }
        else
            t1 += st;
    }

    x_res_file.close();
    x_exact_res_file.close();

    return 0;
}
```

Перейдем к вычислениям с помощью составленной программы. Как говорилось выше, поскольку рассматриваемый случай гармонической внешней силы допускает аналитическое решение, то мы имеем возможность непосредственно оценить точность работы численного метода. Для этого мы в функции `main` выводим с некоторым интервалом времени (равном 1 в приведенном тексте) максимальный модуль отклонения численного решения от аналитического к данному моменту. Запуская программу с некоторыми "разумными" начальными данными и параметрами, получим на экране (в случае приведенного кода):

```
time: 1   error: 1.56344e-007
time: 2   error: 2.45403e-007
time: 3   error: 2.49116e-007
time: 4   error: 6.93856e-007
time: 5   error: 6.97645e-007
time: 6   error: 8.89064e-007
time: 7   error: 1.28346e-006
```

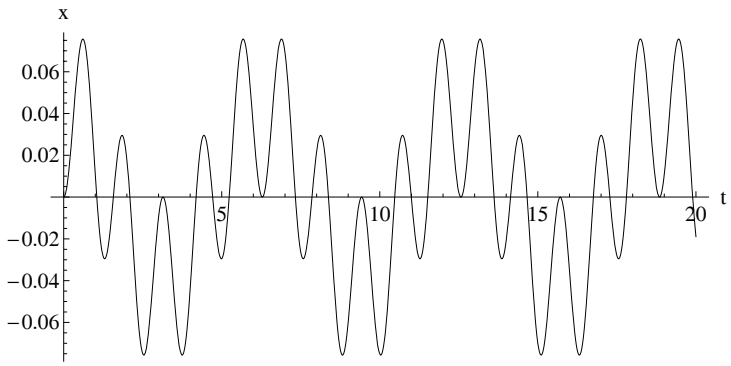


Рис. 2.1. Отсутствие резонанса, наложение колебаний: эволюция координаты частицы $x(t)$ при отсутствии трения, при сильно различающихся собственной и внешней частотах, $\gamma = 0$, $\omega_0 = 1$, $\omega = 5$, $x_0 = 0$, $\dot{x}_0 = 0$.

```
time: 8   error: 1.28346e-006
. . .
time: 200   error: 4.40115e-006
```

Таким образом, при выбранном шаге интегрирования 0.001 сохраняются пять знаков после запятой в величине `x`, которая сама порядка единицы. Если уменьшить шаг в 10 раз, то есть задать его равным 0.0001, то получим:

```
time: 1   error: 2.65392e-010
time: 2   error: 8.09929e-010
time: 3   error: 1.30159e-009
. . .
time: 200   error: 4.67997e-008
```

Видно, что ошибка уменьшилась примерно в 100 раз. Это согласуется с тем, что формула трапеций имеет второй порядок точности. Таким образом, в данном случае с помощью метода трапеций можно обеспечить приемлемую точность вычислений.

2.2.4. Исследование задачи с помощью программы

Приступим к обсуждению характера динамики, возникающей в нашей системе в различных ситуациях. Сначала рассмотрим случай отсутствия трения: $\gamma = 0$. При этом возьмем собственную частоту осциллятора $\omega_0 = 1$, а частоту внешней силы $\omega = 5$, то есть зададим их далекими друг от друга. Соответствующий график приведен на рис. 2.1. Видим, что динамика представляет собой суперпозицию двух колебаний: резкие зигзаги соответствуют колебанию с высокой частотой, а воображаемая более плавная кривая, на которую они "нанизаны" – колебанию с меньшей частотой. Этот вывод можно подтвердить с помощью формулы точного решения. Действительно, в отсутствие трения (и при нулевых начальных координате и скорости) оно записывается в виде (см. (2.8)):

$$x(t) = \frac{F}{\omega_0^2 - \omega^2} \left(\cos \omega t - \cos \omega_0 t\right), \tag{2.11}$$

что и дает два колебания с периодами $T_{\omega_0} = \dfrac{2\pi}{\omega_0} \approx 6.3$ и $T_\omega = \dfrac{2\pi}{\omega} \approx 1.2$.

Зададим теперь значения частот, более близкие друг к другу: $\omega_0 = 1, \omega = 1.2$. График показан на рис. 2.2. Динамика приобретает характер колебания с медленно меняющейся амплитудой (также совершающей колебания). Такой процесс называют биениями. Аналитически его проще всего рассматривать, преобразовав в решении (2.11) произведение в сумму:

$$x(t) = -\frac{2F}{\omega_0^2 - \omega^2} \sin \frac{\varepsilon t}{2} \sin \left(\omega_0 + \frac{\varepsilon}{2}\right)t,$$

где $\varepsilon = \omega - \omega_0$ – отклонение внешней частоты от собственной частоты осциллятора. Видно, что имеется колебание с частотой $\omega_0 + \dfrac{\varepsilon}{2}$, амплитуда которого колеблется с малой частотой $\dfrac{\varepsilon}{2}$. При этом максимальная амплитуда $-\dfrac{2F}{\omega_0^2 - \omega^2} \approx \dfrac{F}{\omega_0 \varepsilon}$ обратно пропорциональна отклонению от резонанса.

Рассмотрим теперь случай полного (точного) резонанса: $\omega_0 = \omega = 1$. Имеем график, изображенный на рис. 2.3. Происхо-

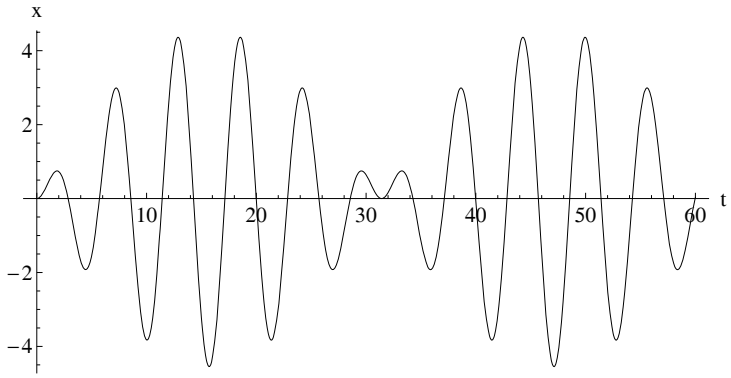


Рис. 2.2. Биения: эволюция координаты частицы $x(t)$ при отсутствии трения, вблизи резонанса, т. е. при близких собственной и внешней частотах, $\gamma = 0$, $\omega_0 = 1$, $\omega = 1.2$, $x_0 = 0$, $\dot{x}_0 = 0$.

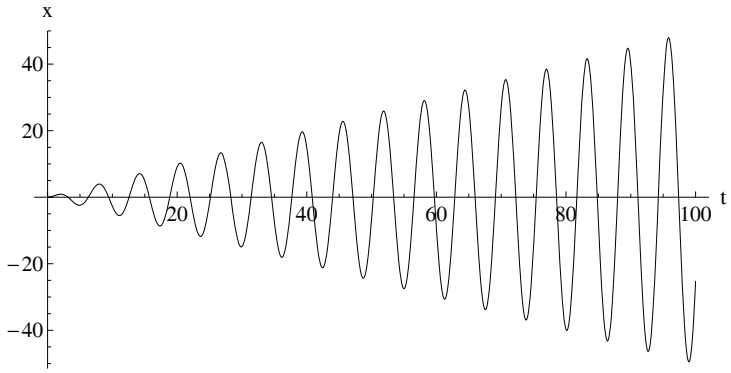


Рис. 2.3. Резонанс: эволюция координаты частицы $x(t)$ при отсутствии трения, в точном резонансе, т. е. при равных собственной и внешней частотах, $\gamma = 0$, $\omega_0 = 1$, $\omega = 1$, $x_0 = 0$, $\dot{x}_0 = 0$.

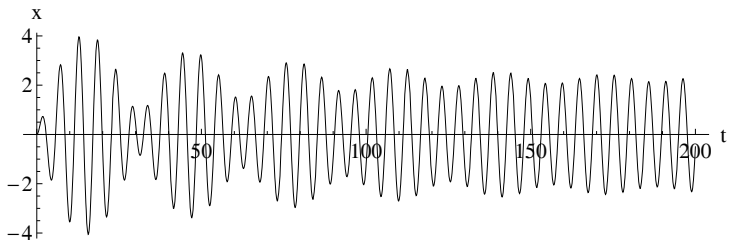


Рис. 2.4. Установление колебаний: эволюция координаты частицы $x(t)$ при наличии трения, при $\varepsilon > \gamma, \gamma = 0.03, \omega_0 = 1, \omega = 1.2$, $x_0 = 0$, $\dot{x}_0 = 0$.

дит колебание с линейно растущей амплитудой. Аналитически это соответствует резонансному решению уравнения колебаний без трения:

$$\ddot{x} + \omega_0^2 x = F \cos \omega t, \tag{2.12}$$

то есть решению для $\omega = \omega_0$. Как известно, в этом случае следует искать частное решение в виде

$$x(t) = t\,(p \cos \omega t + q \sin \omega t)$$

(степень 1, в которой входит переменная t, равна кратности корня ω в характеристическом многочлене линейного уравнения). Подставляя в уравнение движения (2.12), находим:

$$p = 0, \quad q = \frac{F}{2\omega_0}.$$

Отсюда частное решение:

$$x(t) = \frac{F}{2\omega_0} t \sin \omega t,$$

которому вполне соответствует приведенный график и его интерпретация. Таким образом, в точном резонансе происходит неограниченный рост амплитуды колебаний по линейному закону.

Обратимся теперь к случаю ненулевого трения: $\gamma \neq 0$. Снова введем величину, показывающую отклонение от резонанса:

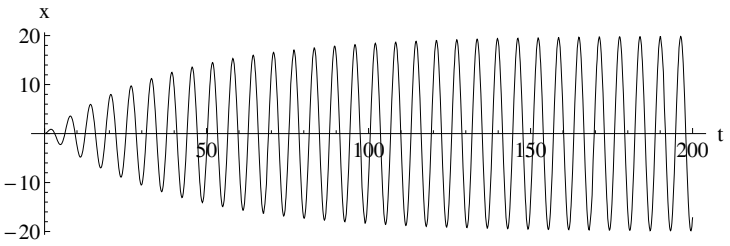


Рис. 2.5. Установление колебаний: эволюция координаты частицы $x(t)$ при наличии трения, при $\varepsilon \ll \gamma$, $\gamma = 0.05$, $\omega_0 = 1$, $\omega = 1.0001$, $x_0 = 0$, $\dot{x}_0 = 0$.

$\varepsilon = \omega - \Omega$ (напомним, что $\Omega = \sqrt{\omega_0^2 - \frac{\gamma^2}{4}}$ – частота собственных колебаний осциллятора). Сначала рассмотрим случай, когда $\varepsilon > \gamma$, например, $\omega_0 = 1, \omega = 1.2, \gamma = 0.03$. График приведен на рис. 2.4. Видно, что происходят колебания, амплитуда которых, в свою очередь, осциллирует, и ее осцилляции постепенно затухают. Процесс приближается к режиму обычных гармонических колебаний, которые в этом случае называются установившимися колебаниями. Изображенную динамику называют переходным процессом, или процессом установления колебаний.

Теперь рассмотрим случай, когда трение "превалирует" над удалением от резонанса: $\varepsilon \ll \gamma$, например, $\omega_0 = 1, \omega = 1.0001, \gamma = 0.05$. Получим траекторию, показанную на рис. 2.5. Здесь мы имеем переходный процесс с плавно возрастающей амплитудой, которая стремится к некоторой константе – амплитуде установившихся колебаний.

Таким образом, при наличии трения динамика содержит переходный процесс, на протяжении которого затухают вызванные внешней силой свободные колебания (на собственной частоте осциллятора), а также последующие установившиеся вынужденные колебания. Продолжительность переходного процесса имеет порядок времени затухания экспоненты $e^{-\gamma t}$, т. е. порядка $\frac{1}{\gamma}$. Амплитуда установившихся колебаний равна $\frac{F}{2\omega_0\varepsilon}$, то есть тем больше, чем ближе частота внешней силы к собственной частоте осциллятора.

2.2.5. Выводы

Явление силового резонанса является важнейшим проявлением действия внешних сил на колебательные системы. Через него устанавливается соотношение между частотой воздействующей на осциллятор силы и собственной частотой осциллятора: как мы видели, резонанс, то есть существенное возрастание амплитуды колебаний, происходит, когда эти частоты близки друг к другу. Иначе говоря, система сильно поглощает энергию тех воздействий, которые происходят на близких к ее собственным частотах, и пропускает другие воздействия. Это свойство имеет широчайшие проявления в природе и применения в технике. Так, оно лежит в основе радиотехники, акустических приборов и музыкальных инструментов, лазерной техники. Более экзотические примеры включают орбитальный резонанс, из-за которого, например, периоды орбит трех из четырех галилеевских спутников Юпитера – Ганимеда, Европы и Ио – относятся как 4:2:1 (так называемый резонанс Лапласа), а также приливный резонанс, приводящий к резкому усилению приливов в бухтах, где их период совпадает со временем прохождения бухты длинной волной. Кроме того, резонансные явления, хотя и несколько своеобразной, квантовой природы, лежат в основе многих методов изучения вещества. В этом случае внешние воздействия резонируют с параметрами систем атомного масштаба. Например, так устроены оптическая и инфракрасная спектроскопии газов, эффект Мёссбауэра, магнитный резонанс. Следует также отметить, что во многих ситуациях резонансы могут иметь нежелательные для человека последствия: разрушение мостов, башен, механических устройств. Все это делает изучение резонансных явлений одной из центральных областей внимания исследователей.

2.3. Параметрический резонанс

Рассмотрим осциллятор при наличии диссипации, на который не действует внешняя сила, но упругая константа которого периодически меняется со временем

$$\ddot{x} + \gamma \dot{x} + (\omega^2 - 2\varepsilon \sin 2\nu t)\, x = 0$$

Попытка применить метод вариации произвольных постоянных ни к чему хорошему не приводит. Обратимся к методу, предложенному Rayleigh'ем [1)]. Ищем решение в виде ряда Фурье по нечётным значениям частоты ν

$$x = A_1\, sin\,\nu t + B_1\, cos\,\nu t + A_3\, sin\, 3\nu t + B_3\, cos\, 3\nu t \\ + A_5\, sin\, 5\nu t + B_5\, cos\, 5\nu t + \ldots$$

Коэффициенты $A_1, B_1, A_3, B_3, A_5, B_5, \ldots$ находим, подставляя привдённое выше выражение в уравнение движения осциллятора и приравнивая коэффициенты при соответствующих $sin\,\nu t, cos\,\nu t, sin\, 3\nu t, cos\, 3\nu t, \ldots$. Поскольку в уравнение движения входит $sin\, 2\nu t$, в разложении будут встречаться только нечётнократные функции от ν. В результате получается система линейных уравнений:

$$\begin{array}{lllll}
A_1(\omega^2-\nu^2) & - & (\gamma\nu+\varepsilon)B_1 & + & \varepsilon B_3 = 0 \\
B_1(\omega^2-\nu^2) & + & (\gamma\nu-\varepsilon)A_1 & - & \varepsilon A_3 = 0 \\
A_3(\omega^2-9\nu^2) & - & (3\gamma\nu+\varepsilon)B_1 & + & \varepsilon B_5 = 0 \\
B_3(\omega^2-9\nu^2) & + & (3\gamma\nu+\varepsilon)A_1 & - & \varepsilon A_5 = 0 \\
A_5(\omega^2-25\nu^2) & - & (5\gamma\nu+\varepsilon)B_5 & + & \varepsilon B_7 = 0 \\
B_5(\omega^2-25\nu^2) & + & (5\gamma\nu+\varepsilon)A_5 & - & \varepsilon A_7 = 0 \\
\ldots\ldots\ldots\ldots & & & &
\end{array}$$

На практике ε – малый параметр, по которому удобно вести разложение. Видим, что $A_3,\ B_3$ порядка ε относительно $A_1,\ B_1$; $A_5,\ B_5$ порядка ε относительно $A_3,\ B_3$ и так далее. Ввиду этого ограничимся членами главного порядка и, опустив $A_3,\ B_3$ в первых двух уравнениях, получим

$$A_1(\omega^2-\nu^2)-B_1(\gamma\nu+\varepsilon)=0, \qquad B_1(\omega^2-\nu^2)+A_1(\gamma\nu-\varepsilon)=0.$$

Условием согласованности этих уравнений является

$$(\omega^2-\nu^2)^2 = \varepsilon^2 - \gamma^2\,\nu^2 \tag{2.13}$$

из которого следует существование порога возбуждения

$$\varepsilon^2 - \gamma^2\,\nu^2 \geq 0 \tag{2.14}$$

[1)] J.W.Strutt, Baron Rayleigh, The Theory of Sound, volume I, Ch.3; имеется русский перевод: Дж.Рэлей, Теория звука, том I, ГИТТЛ, Москва-Ленинград (1940).

Кроме того, имеем соотношение

$$\frac{A_1}{B_1} = \frac{\gamma\,\nu + \varepsilon}{\omega^2 - \nu^2} = \sqrt{\frac{\varepsilon + \gamma\nu}{\varepsilon - \gamma\nu}}$$

Из линейности исходного уравнения следует, что по достижении порога амплитуда колебания ничем не ограничена. Стоит отметить, что при параметрическом резонансе имеет место трансформация волн из области болеее высоких частот в область более низких, $2\,\nu \Rightarrow \omega \approx \nu$, что даёт ряд преимуществ при применении его в технике [1)].

2.4. Задача для компьютерного моделирования: параметрический резонанс для осциллятора с диссипацией

2.4.1. Мотивировки и постановка задачи

Каждый, кто качался на качелях, интуитивно представляет себе, как надо двигаться, чтобы от первоначально небольшой амплитуды раскачаться до внушительного размаха. Если подумать об этом более вимательно, можно заметить, что нужно двигаться в такт качелям: опускаться (приседать), когда они движутся вниз, и подниматься – когда наверх. Так, интуитивно, люди в раннем детстве знакомятся с важным явлением теории колебаний – параметрическим резонансом. Суть явления состоит в следующем. Раскачиваясь на качелях, мы то приседаем, то выпрямляемся, то есть меняем высоту своего центра масс. Поскольку, как известно, период колебаний маятника возрастает с увеличением длины подвеса, или, в более общем случае, с увеличением расстояния от точки подвеса до центра масс маятника, то получается, что мы изменяем тем самым период или, что тоже, частоту колебаний качелей периодическим образом. Так как за один период колебаний качелей мы два раза

[1)] Теория параметрического резонанаса была создана G. Hill'ом при исследовании движения Луны, см. G.Hill, "On the Part of the Motion of the Lunar Pedigree, which is a Function of the Mean Motion of the Sun and the Moon Acta Mathematica, **8** , 1 (1886).

проходим нижнее положение, а значит, совершаем два приседания, то получаем, что частота нашего изменения длины подвеса вдвое больше частоты колебаний качелей. Как было показано в предыдущем разделе, к такому результату (в нулевом приближении) приводит и теоретическое рассмотрение.

Промоделируем численно решения уравнения

$$\ddot{x} + \gamma\,\dot{x} + (\omega^2 - 2\,\varepsilon\,\sin 2\nu t)\,x = 0$$

осциллятора с трением, с периодически меняющейся частотой.

2.4.2. Программная реализация

Очевидно, структура программы для этой задачи вполне аналогична той, которая использовалась при изучении силового резонанса. Разница сводится к изменению вида решаемого уравнения, а также к отсутствию в данном случае точных решений.

Итак, текст программы выглядит следующим образом.

Заголовочный файл `"par_res.cpp"`, содержащий определения пользовательских типов данных и объявления функций:

```
#ifndef PAR_RES_H
#define PAR_RES_H

struct Var   // переменные системы первого порядка
{
    double x, v;
};

// арифметические операции над структурами переменных:
Var operator +(Var& A, Var& B);
Var operator *(double k, Var& A);

struct Par   // структура параметров задачи
{
    double omega, gamma, eps, nu;
};
```

```
// функция, вычисляющая правые части
// системы первого порядка:
Var RHS(Var& V, Par& P, double t);

// шаг метода трапеций:
void Trapezium_step(Var& V, Par& P, double& t,
                                          double st);

#endif
```

Файл "var.cpp", в котором определяются алгебраические операции над структурами Var:

```
#include "par_res.h"

Var operator +(Var& A, Var& B)
{   // сложение двух структур переменных
    Var Res;
    Res.x = A.x + B.x;
    Res.v = A.v + B.v;
    return Res;
}

Var operator *(double k, Var& A)
{   // умножение структуры переменных на число
    Var Res;
    Res.x = k * A.x;
    Res.v = k * A.v;
    return Res;
}
```

Файл "integ.cpp", содержащий функцию шага метода интегрирования:

```
#include "par_res.h"
#include "math.h"

void Trapezium_step(Var& V, Par& P, double& t,
                                          double st)
{
    Var V0 = V;   // структура переменных
```

```
                // на предыдущем шаге
    Var F0 = RHS(V0, P, t);   // значение правых частей
                              // на предыдущем шаге
    Var V_prev = V;
    // вспомогательная структура, хранящая значение
    // вычисляемой новой структуры переменных,
    // полученное на предыдущей итерации

    do   // цикл решения алгебраического уравнения
    {     // методом простых итераций
        V_prev = V;

        V = V0 + 0.5 * st * ( F0
             + RHS(V_prev, P, t + st) );
    }
    while ( fabsl(V.x - V_prev.x)
          + fabsl(V.v - V_prev.v) > 0.00000001 );
  // сравнение результатов текущей и предыдущей
  // итераций в норме, равной сумме модулей компонент;
  // вычисление останавливается, если норма разности
  // меньше заданного значения

    t += st;   // сдвиг текущего времени на шаг
}
```

Файл "par_res.cpp", содержащий функцию правых частей:

```
#include "par_res.h"
#include <math.h>

Var RHS(Var& V, Par& P, double t)
{   // правые части системы первого порядка
    Var Res;
    Res.x = V.v;
    Res.v = - P.gamma * V.v - pow(P.omega, 2)
     * (1. - 2. * P.eps * sinl(2. * P.nu * t) ) * V.x;
    return Res;
}
```

Наконец, файл "main.cpp", содержащий управляющую функцию:

```
#include "par_res.h"
#include <math.h>
#include <fstream>
#include <iostream>

using namespace std;

int main()
{
    Var V;
    V.x = 0.;   V.v = .1;
    Var V0 = V;
    Par P;
    P.omega = 1.;   P.gamma = 0.01;
    P.eps = 0.03;   P.nu = 1.;
    double t = 0.;
    double st = 0.001;

    ofstream x_res_file("x.res");
    // файл численной траектории

    double t1 = 0;
    double point_out_interval = 0.01;
    // вспомогательные переменные
    // для вывода в файл значений x
    // с интервалом времени point_out_interval

    x_res_file << V.x << " ";

    while (t < 500)   // рассчитываем траекторию
    {    // на интервале времени (0, 500)
        Trapezium_step(V, P, t, st);
        // шаг интегрирования

        t1 += st;
        if ( t1 > point_out_interval - st / 2.)
        {   //вывод численного решения в файл
            x_res_file << V.x << " ";
```

```
            t1 = 0;
        }
    }

    x_res_file.close();
    return 0;
}
```

2.4.3. Исследование задачи с помощью программы

Зададим частоту параметрического возмущения $2\,\nu$ равной удвоенной частоте ω осциллятора, то есть положим $\nu = \omega$. Кроме того, легко понять, что параметрический резонанс, в отличие от силового, может происходить только при ненулевых начальных условиях, так как задание $x(0) = 0$, $\dot{x}(0) = 0$ приводит, очевидно, к тождественно нулевому решению (уравнение однородно). Поэтому зададим, например, $x(0) = 0$, $\dot{x}(0) = 0.1$. При значениях коэффициента диссипации (трения) $\gamma = 0.01$, а параметра величины возмущения $\varepsilon = 0.03$ имеем решение, показанное на рис. 2.6. Видим, что амплитуда колебаний неограниченно растет[1)].

В приведенном примере было выполнено условие (2.14), задающее порог для величины возмущения ε, превышение которого необходимо для существования резонанса:

$$|\varepsilon| \geqslant \gamma\,\nu.$$

Если же оно не выполнено, например, при слишком большом (относительно других величин) трении: $\gamma = 0.05$, а остальные параметры, как и раньше: $\varepsilon = 0.03$, $\nu = 1$, $\omega = 1$, то получим решение, изображенное на рис. 2.7. Видно, что резонанс отсутствует, он подавлен трением.

Удаление частоты параметрического возмущения ν от резонансного значения ω также приводит к исчезновению резонанса. Например, в случае $\nu = 1.1$, $\omega = 1$, $\gamma = 0.01$, $\varepsilon = 0.03$ получаем результат, показанный на рис. 2.8.

Даже задание нулевого трения не приводит к появлению резонанса: при значениях $\nu = 1.1$, $\omega = 1$, $\gamma = 0$, $\varepsilon = 0.03$ имеем

[1)] Можно показать, что этот рост происходит по экспоненте.

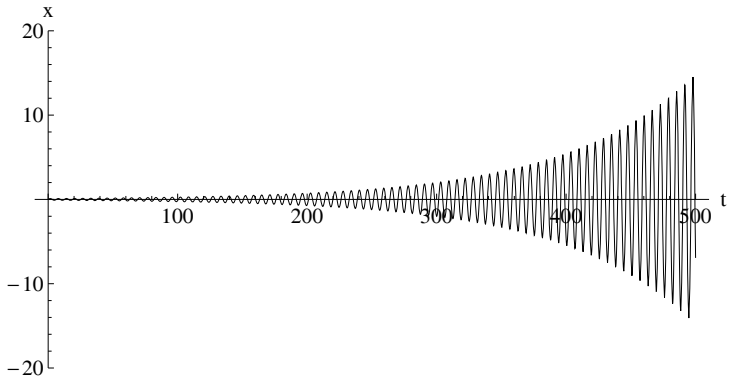


Рис. 2.6. Параметрический резонанс, $\omega = 1$, $\nu = 1$, $\gamma = 0.01$, $\varepsilon = 0.03$.

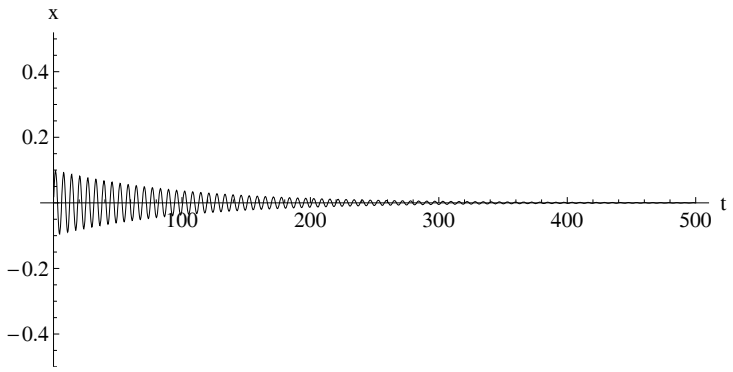


Рис. 2.7. Подавление резонанса трением, $\omega = 1$, $\nu = 1$, $\gamma = 0.05$, $\varepsilon = 0.03$.

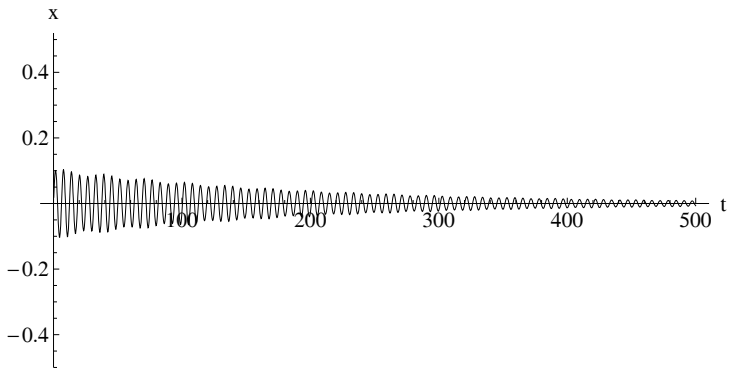


Рис. 2.8. Отклонение параметрической частоты ν от резонансного значения, равного ω, $\omega = 1$, $\nu = 1.1$, $\gamma = 0.01$, $\varepsilon = 0.03$.

решение, приведенное на рис. 2.9.

2.4.4. Выводы

Итак, мы рассмотрели поведение осциллятора с периодически меняющейся частотой. Как и предсказывает теория, наблюдается эффект параметрического резонанса при частотах колебания параметра, близких к удвоенной частоте самого осциллятора. Эффект ослабляется при увеличении трения γ, уменьшении величины параметрических колебаний ε и при отклонении параметрической частоты от резонансного значения.

Параметрический резонанс имеет применения в электротехнике. Кроме того, оказывается, что уравнение, подобное рассмотренному выше, описывает поведение квантовой частицы в периодическом потенциале. Поэтому условия, когда его решения ограничены, или, наоборот, их амплитуда неограниченно растет, играют ключевую роль в физике твердого тела, составляя основу так называемой зонной теории кристаллов.

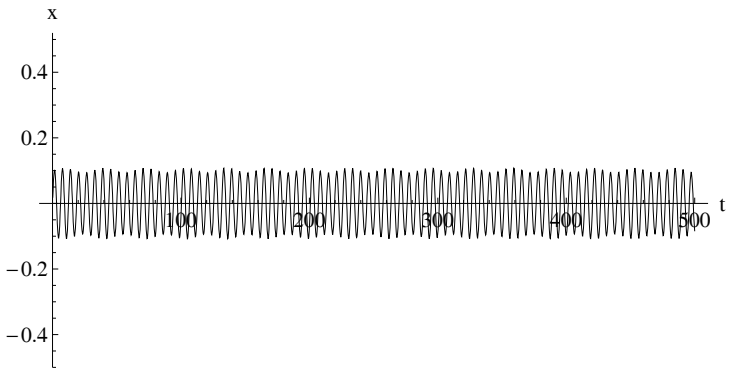


Рис. 2.9. Отклонение параметрической частоты ν от резонансного значения, равного ω, случай нулевого трения, $\omega = 1$, $\nu = 1.1$, $\gamma = 0$, $\varepsilon = 0.03$.

Глава 3

Задача Кеплера

Задача Кеплера о движении материальной точки под действием силы, притягивающей тело по закону $\propto 1/r^2$, играет совершенно особую роль в точных науках. Исторически она появилась впервые в связи с теорией тяготения Newton'a, но оказалось, что та же зависимость имеет место и в теории электричества, где она описывает притяжение – отталкивание зарядов (закон Кулона). Далее её победное шествие распространилось на квантовую механику, одним из первых успехов которой было квантование атома водорода, т.е. точное решение квантовой задачи Кеплера. В основе этих успехов лежат особые свойства кулоновского потенциала $1/r$, обладающего целым рядом очевидных – геометрических – симметрий, а также скрытыми, или динамическими. Как мы увидим на примере этой задачи, имеющиеся у нас простые средства позволяют, тем не менее, получать инересные результаты.

Будем рассматривать движение материальной точки массы m в поле притягивающего центра. Предположим, что тело, оказывающее притяжение, является точечным и его положение фиксировано в начале координат. Последнее можно принять, если, например, масса притягивающего тела много больше, чем масса первого тела. Будем считать, что взаимодействие тела с притягивающим центром происходит по закону

$$\vec{f} = -\frac{\gamma m\, M}{r^3}\,\vec{r} \tag{3.1}$$

где $\vec{r}$ – вектор из начала координат в точку нахождения тела, m – масса тела, M – масса притягивающего центра, γ – коэффициент пропорциональности, или гравитационная постоянная. Уравнение (3.1) соответствует закону всемирного тяготения Ньютона. Аналогичное уравнение имеет место для электростатического взаимодействия электрических зарядов, закон Кулона; в последнем случае надо вместо масс написать электри-

ческие заряды центра и частицы $Q,\ q$ и вместо гравитационной постоянной $1/\varepsilon$, где ε – диэлектрическая проницаемость. Нас в дальнейшем будут интересовать формальные свойства динамики частицы.

3.1. Сохранение энергии и углового момента

Второй закон Ньютона в данном случае имеет вид

$$m\,\frac{d^2}{dt^2}\vec{r} = -\frac{\gamma m\,M}{r^3}\,\vec{r} \tag{3.2}$$

Умножим обе части на скорость тела и, проинтегрировав, получим

$$\frac{m\,\vec{v}^{\,2}}{2} - \frac{\gamma\,mM}{r} = E = const \tag{3.3}$$

Первое слагаемое есть кинетическая энергия тела, второе – потенциальная энергия

$$U = -\frac{\gamma\,mM}{r}$$

которая связана с силой по формуле

$$\vec{f} = -\frac{\partial U}{\partial \vec{r}}$$

Таким образом, уравнение (3.3) есть закон сохранения энергии.

Рассмотрим выражение

$$\vec{L} = \vec{r} \times \vec{p} \qquad (\vec{p} = m\,\vec{v}) \tag{3.4}$$

– угловой момент, или момент количества движения. Продифференцировав его по времени и воспользовавшись (3.2), получаем

$$\frac{d}{dt}\vec{L} = 0$$

т.е. имеет место сохранение углового момента, $\vec{L} = const$.

Из сохранения углового момента следует, что траектория тела – плоская кривая. Чтобы убедиться в этом, умножим постоянный вектор $\vec{L}$ на радиус-вектор $\vec{r}$. Получим

$$\vec{v} \cdot \vec{L} = 0$$

что означает, что траектория лежит в фиксированной плоскости, проходящей через начало координат.

3.2. Редукция к одномерной задаче

Выберем оси координат таким образом, чтобы ось z совпадала с направлением $\vec{L}$. В плоскости $x-y$ введём полярную систему координат, $x = r\cos\phi,\ y = r\sin\phi$. Тогда интеграл энергии E и третьей компоненты углового момента $L_3 = L$ примут вид

$$E = \frac{m}{2}\,(\dot{r}^2 + r^2\dot{\phi}^2) - \frac{\gamma\, mM}{r}, \qquad L = m\, r^2\,\dot{\phi}$$

Условие

$$L = m\, r^2\,\dot{\phi} = const \tag{3.5}$$

есть не что иное, как второй закон Кеплера, закон постоянства секторальных скоростей. Выразим из него $\dot{\phi}$ и подставим в уравнение сохранения энергии. Получим

$$E = \frac{m}{2}\,\dot{r}^2 + \frac{L^2}{2mr^2} - \frac{\gamma mM}{r} \tag{3.6}$$

Уравнение (3.6) представляет собой закон сохранения энергии для одномерного движения с эффективной потенциальной энергией

$$U_{eff} = \frac{L^2}{2mr^2} - \frac{\gamma mM}{r} \tag{3.7}$$

Оно позволяет сделать некоторые заключения о характере орбит.

Мы видим, что при $r \to 0$ эффективный потенциал (3.7) стремится к бесконечности, при $r \to 0$ он стремится к нулю, оставаясь отрицательным, обращается в нуль только в одной точке

$$r_* = \frac{L^2}{2m^2 M\gamma}$$

и имеет только один минимум, в котором он меньше нуля.

Законы сохранения (3.6)(3.5) позволяют написать решение задачи в виде интегралов. Для этого заметим, что (3.5)-(3.6)

можно переписать в виде

$$\begin{aligned} \frac{dr}{dt} &= \sqrt{\frac{2}{m}\left[E - U_{eff}\right]}, \\ \frac{d\phi}{dt} &= \frac{L}{r^2} \end{aligned}$$

Откуда получаем

$$\frac{d\phi}{dr} = \frac{L}{r^2\sqrt{\frac{2}{m}\left[E - U_{eff}\right]}}$$

Последнее уравнение позволяет записать ответ в виде интеграла

$$\phi = \phi_0 + \int \frac{\frac{L}{r^2}}{\sqrt{\frac{2}{m}\left[E - U_{eff}\right]}} dr \tag{3.8}$$

Удобнее, однако, провести исследование решений, пользуясь более простыми средствами.

3.3. Законы Кеплера, типы орбит

Определим геометрический тип траекторий в задаче Кеплера. Для этого воспользуемся полярными координатами. Выберем систему координат, в которой орбита лежит в плоскости $z = 0$. Уравнения движения примут вид

$$\ddot{x} + \frac{\gamma\, M}{r^3}\, x = 0, \qquad \ddot{y} + \frac{\gamma\, M}{r^3}\, y = 0$$

где M – массса притягивающего центра, находящегося в начале координат. Полярные координаты заданы уравнениями

$$x = r\, \cos\, \phi, \qquad y = r\, \sin\, \phi$$

В переменых $r,\ \phi$ уравнения движения принимают вид

$$\ddot{r}\, \cos\, \phi - r\, \sin\phi\ \ddot{\phi} - r\, \cos\phi\, \dot{\phi}^2 - 2\sin\phi\, \dot{r}\, \dot{\phi} + \frac{\gamma\, M}{r^2}\, \cos\phi \quad = \quad 0,$$

$$\ddot{r}\ \sin\ \phi + r\ \cos\phi\ \ \ddot{\phi} - r\ \sin\phi\ \dot{\phi}^2 + 2\cos\phi\ \dot{r}\ \dot{\phi} + \frac{\gamma\, M}{r^2}\ \sin\phi \quad = \quad 0$$

Умножим первое уравнение на $\cos\phi$, а второе на $\sin\phi$ и сложим. Получим

$$\ddot{r} - r\ \dot{\phi}^2 + \frac{\gamma\, M}{r^2} = 0$$

Полученное уравнение совместно с законом сохранения секторальных скоростей (3.5), вторым законом Кеплера, полностью определяет движение. Положим $u = 1/r$. Имеем

$$\dot{r} = \frac{d}{dt}\left(\frac{1}{u}\right) = \frac{d}{d\phi}\left(\frac{1}{u}\right)\ \frac{d\phi}{dt} = -\ \frac{1}{u^2}\frac{du}{d\phi}\ \dot{\phi}$$

И аналогично для второй производной

$$\ddot{r} = -\ L^2\, u^2\, \frac{d^2u}{d\phi^2}$$

В результате получаем уравнение

$$\frac{d^2u}{d\phi^2} + u = \frac{\gamma\, Mm^2}{L^2}$$

которое имеет общее решение

$$u = \frac{\gamma M m^2}{L^2}\ \ (1 + e\ \cos(\phi - \phi_0))$$

или

$$r = \frac{L^2}{\gamma M m^2}\ \frac{1}{1 + e\ \cos(\phi - \phi_0)}$$

Видим, что траектория орбиты является кривой второго порядка: при $e < \ 1, e\ > 1, e = 1$ – эллипс, гипербола или парабола соответственно.

Рассмотрим случай эллиптической орбиты и найдём соотношение между величиной большой полуоси и периодом обращения – третий закон Кеплера. Для этого заметим, что секторальная скорость связана с производной элемента площади A, заметаемой радиус-вектором из притягивающего центра в точку орбиты, по формуле

$$\frac{dA}{dt} = \frac{r^2\, \dot{\phi}}{2} = \frac{L}{2m}$$

откуда следует, что период обращения связан с площадью эллипса по формуле

$$A = \frac{L\,T}{2m}$$

Площадь эллипса, как известно, равна $A = \pi\,ab$, где a, b – большая и малые полуоси соответственно. Используя выражения для большой и малой полуоси через фокальный параметр p

$$b^2 = p\,a, \qquad p = \frac{L^2}{\gamma\,M\,m^2}$$

получаем в результате период обращения как функцию величины большой оси – третий закон Кеплера

$$T = 2\pi\,a^{3/2}/\sqrt{\gamma\,M}, \qquad \text{или} \qquad \frac{T^2}{a^3} = \frac{4\pi^2}{\gamma\,M} \tag{3.9}$$

здесь M – масса притягивающего тела ("Солнца"). Правая часть не завивсит от характеристик индивидуальных планет, и, таким образом, получаем третий закон Кеплера

$$\frac{T_1^2}{a_1^3} = \frac{T_2^2}{a_2^3}$$

в его традиционной форме.

3.4. Соображения размерности и подобия. Грубые оценки в физике

Точную (в рамках определенной модели) информацию о физическом явлении дает решение описывающих его уравнений. Однако, за исключением ограниченного множества сравнительно простых задач, получить точные решения в обозримом виде затруднительно, а часто просто невозможно. Кроме того, во многих случаях существуют проблемы на этапе постоения модели физического явления, когда есть трудности с математическим описанием участвующих в нем факторов и соответственно с написанием уравнений, его описывающих. Наконец, нередко встречается ситуация, когда процесс полного решения задачи в

принципе ясен, однако технически трудоемок и займет значительное время, а вопрос, на который нужно ответить, не требует полного описания системы во всех подробностях.

Во всех этих случаях полезно применять оценки по порядку величины, грубые предварительные прикидки, а также метод размерностей. Рассмотрим подробнее последний метод, дающий весьма практичные результаты, если понимать его ограниченность. Этот метод часто употребляется для предварительной оценки характера возможных решений. В целом ряде случаев он позволяет получить окончательный ответ[1)].

Рассмотрим конкретный простой пример – математический маятник. Имеем точечную массу m, подвешенную на невесомом стержне. Стержень с подвешенной массой совершает малые колебания, т.е. колебания с амплитудой много меньше длины подвеса, под действием силы тяжести. Желаем выразить период колебаний T как функцию от остальных величин, описывающих систему. Прежде всего учтём, что в нашем распоряжении имеются только следующие величины, которые существенны для рассматриваемого явления: длина подвеса l, масса частицы m, ускорение силы тяжести g, период колебаний T. Мы исходим, таким образом, из картины явления, в которой тело совершает колебания с фиксированной частотой, или периодом, пренебрегаем сопротивлением воздуха и считаем, что амплитуда не влияет на период (последнее предположение в случае малых колебаний подтверждается экспериментально). Как обычно, воспользуемся системой единиц CGS. Будем искать зависимость вида

$$T = const\ m^a\, g^b\, l^c$$

Константа – безразмерна. Размерности остальных величин

$$[m] = gr, \quad [g] = \frac{cm}{sec^2}, \quad [l] = cm, \quad [T] = sec$$

Сравнивая размерности левой и правой части, получаем урав-

[1)] Как писал Rayleigh, "... метод размерностей несколько опасен; если, однако, применять его с должной осторожностью, то он неоспоримо обладает большой силой и ценностью" , см. Дж. Рэлей, "Теория звука" , ГИТТЛ, Москва – Ленинград (1940).

нения

$$\begin{aligned} a &= 0 \\ c + b &= 0 \\ 2\,b &= 1 \end{aligned}$$

Получаем в результате формулу

$$T = const\,\sqrt{\frac{l}{g}}$$

Значение константы в точном решении равно $2\,\pi$, т.е. качественно ответ вполне удовлетворителен. Конечно, можно возразить, что выбранная изначально форма решения – спорна, что можно было бы ожидать появления вместо *const* некоторой достаточно сложной функции от безразмерной комбинации l и амплитуды[1)], и т.д. Всё это так. Но вспомним замечание Альберта Эйнштейна, что "Господь Бог очень сложен, но не злонамерен"[2)].

Обратимся опять к орбитам планет и выведем из соображений размерности третий закон Кеплера. Обратим внимание на то, что уравнения движения (3.2) инвариантны относительно преобразований изменения масштабов пространственных координат и времени

$$\vec{r} \;\to\; a\,\vec{r}, \qquad t \;\to\; b\,t \tag{3.10}$$

только при выполнении условия

$$a\,b^{\,-2} = a^{\,-2}, \qquad \text{или} \qquad a^3 = b^2$$

При выполнении этих условий преобразования (3.10) переводят решения в решения. Изменение масштаба по пространственным координатам соответствует переходу к подобным фигурам, поэтому из указанного соотношения заключаем, что кубы характерных размеров *геометрически подобных орбит*[3)] должны относиться как квадраты характерных времен их движения.

[1)] В действительности при больших амплитудах именно это имеет место.

[2)] "Raffiniert ist der Herr Gott, aber bösartigt ist er nicht".

[3)] Приведённое рассуждение не позволяет сравнивать орбиты, которые не являются геометрически подобными, хотя сам третий закон Кеплера остаётся верным. Его вывод нуждается в более подробном рассмотрении движения по орбите, см. стр. 78.

Сравнения масштабов играют основополагающую роль при решении физических задач. Можно утверждать, что главное для понимания сущности конкретной проблемы и основной этап в её решении – это выяснение характерных размеров. Эта часть работы обычно не связана с какими-то трудными вычислениями и требует прежде всего внимательного анализа общей ситуации и физической интуиции. Это именно то обстоятельство, в котором яснее всего проявляется различие мышления и психологии физика и математика.

Приведём пример грубой оценки, которая очень часто встречается. Рассмотрим тело линейного размера L. Можно считать, что объём тела растёт как L^3, с другой стороны, площадь его поверхности растёт как L^2. При $L \to \infty$ и умеренных значениях плотности внутренней энергии и поверхностной энергии можно пренебречь поверхностными эффектами.

Вот ещё пример уже из самой математики. Имеется сфера радиуса $R > 1$ в пространстве очень большой размерности N. Её объём будет в основном сосредоточен в узком слое около её поверхности, причём толщина этого слоя будет стремиться к нулю по мере роста $N \to \infty$.

3.5. Оценка Ньютона расстояния до неподвижных звёзд

Пример применения оценки по порядку величин в сочетании с правдоподобным рассуждением, позволившим сделать оценки расстояний в нашей галактике, принадлежит Ньютону. Идея проводимой оценки основана на использовании понятия освещённости – светового потока на единицу площади поверхности. Согласно этому определению освещённость от точечного источника света убывает пропорционально квадрату расстояния. Ньютон воспользовался тем обстоятельством, что освещённость, создаваемая планетой Сатурн, близка к освещённости от некоторых звёзд. Сатурн светит отражённым светом Солнца. Ньютон предположил, что звёзды примерно такие же, как Солнце. Теперь процитируем самого Ньютона[1)]:

[1)] Цитата взята из Ф.Крауфорд, Берклеевский курс физики, том III, Волны, стр. 201, Наука, Москва (1976).

"На диск Сатурна падает около 1 / 2100 000 000 части солнечного света[2)]. Во столько же раз поверхность этого диска меньше сферической поверхности, радиус которой равен радиусу орбиты Сатурна. Предположим далее, что Сатурн отражает 1 / 4 этого света. Тогда отражённый полусферой Сатурна свет будет составлять 1 / 4200 000 000 часть света, испущенного полусферой Солнца.

Уменьшение света обратно пропорционально второй степени расстояния до светящегося тела. Поэтому, если бы Солнце было на расстоянии от Земли в $10000\sqrt{42}$ раз большем, чем Сатурн, оно показалось бы таким же ярким, каким кажется Сатурн без своего кольца. Такое свечение немного превосходило бы свечение неподвижной звезды первой величины. Таким образом, расстояние, с которого Солнце светило бы, как неподвижная звезда, приблизительно в 100 000 раз больше расстояния до Сатурна. При этом его угловой диаметр был бы равен $7^{IV}\,16^{V}$, а параллакс, созданный годичным движением Земли, составил бы 13^{III}. Таковы расстояния, угловой диаметр и параллакс звезд первой величины, которые равны нашему Солнцу по величине и свету".

Расстояния до ближайших звёзд действительно находятся в пределах 20 000 - 100 000 расстояний от Солнца до Сатурна.

3.6. Учёт малых поправок и к каким важным выводам он может приводить

До сих пор мы не задавались вопросом о роли величины рассматриваемых членов. Это, в частности, проявилось в том, что в кеплеровой задаче мы считали притягивающий центр неподвижным, в то время как в реальной жизни, будет ли рассматриваемая задача из небесной механики или электростатики, притягивающий центр, хоть немного, но смещается. Предположение, что он неподвижен, основывалось на том, что его масса очень велика. Попытаемся учесть соответствующую поправку, учитывая, что у нас имеется малый параметр m/M –

[2)] Диаметр Сатурна $1.21 \cdot 10^5\ km$, его среднее расстояние от Солнца $1.43 \cdot 10^9\ km$, поэтому Сатурн, рассматриваемый с Солнца, занимает 1 / 2 240 000 000 часть сферы.

отношение массы рассматриваемого тела m к массе притягивающего центра M.

Будем предполагать, что оба тела (тело с малой массой будем называть спутником) двигаются таким образом, что расстояние между телом и спутником много меньше расстояния до притягивающего центра – Солнца. Возьмём систему координат с центром в притягивающем теле. Обозначим через $\vec{r}$ вектор из начала координат в точку нахождения тела, через $\vec{r}_1$ – в точку нахождения спутника. Уравнения движения для тела и спутника имеют вид

$$m\,\frac{d^2}{dt^2}\,\vec{r} \;=\; -\,\gamma\,\frac{m\,M}{|\vec{r}|^{3/2}}\,\vec{r} - \;\gamma\,\frac{m\,m_1}{|\vec{r}-\vec{r}_1|^{3/2}}\,(\vec{r}-\vec{r}_1) \qquad (3.11)$$

$$m_1\,\frac{d^2}{dt^2}\,\vec{r}_1 \;=\; -\,\gamma\,\frac{m_1\,M}{|\vec{r}_1|^{3/2}}\,\vec{r}_1 - \;\gamma\,\frac{m_1\,m}{|\vec{r}-\vec{r}_1|^{3/2}}\,(\vec{r}_1-\vec{r}) \qquad (3.12)$$

где m – масса тела, m_1 – его спутника.

Разделим уравнение (3.11) на m, а уравнение (3.12) на m_1 и вычтем второе из первого, получим

$$\frac{d^2}{dt^2}\,(\vec{r}-\vec{r}_1) = -\,\gamma\,\frac{m+m_1}{|\vec{r}-\vec{r}_1|^{3/2}}\,(\vec{r}-\vec{r}_1) - \;\gamma\,M\,\left(\frac{1}{|\vec{r}|^{3/2}} - \frac{1}{|\vec{r}_1|^{3/2}}\right) \qquad (3.13)$$

Поскольку $|\vec{r}-\vec{r}_1| \ll |\vec{r}|, |\vec{r}_1|$, отбросим второе слагаемое в правой части уравнения (3.13) и положим $\vec{\xi} = \vec{r} - \vec{r}_1$. В результате получим

$$\frac{d^2}{dt^2}\,\vec{\xi} = -\,\gamma\,\frac{m+m_1}{|\vec{\xi}|^{3/2}}\,\vec{\xi} \qquad (3.14)$$

Применяя третий закон Kepler'a (3.9), можем записать

$$m + m_1 = 4\pi^2\frac{a^3}{\gamma\,T^2}$$

Для пары притягивающий центр (Солнце) – планета имеем третий закон Кеплера

$$M + m = 4\pi^2\frac{a^3_{\text{планета}}}{\gamma\,T^2_{\text{планета}}}$$

Разделив второе уравнение на первое и учтя, что масса спутника мала, получим

$$\frac{M+m}{m} = \left(\frac{a_{\text{спутник}}}{a}\right)^3 \left(\frac{T}{T_{\text{спутник}}}\right)^2 \qquad (3.15)$$

Исходя из астрономических наблюдений можно найти отношения $a/a_{\text{планета}}$, $T/T_{\text{планета}}$, и с помощью уравнения (3.15) найти отношение массы планеты к массе Солнца.

3.7. Задача для компьютерного моделирования: задача Кеплера

3.7.1. Мотивировки и постановка задачи

Задача Кеплера, как было показано выше, эффективно решается аналитическими методами. Это один из тех немногочисленных примеров, в которых методики точного решения дифференциальных уравнений достигают полного успеха, приводя к простому и конструктивному ответу. Такие задачи играют огромную роль в механике и в физике вообще, так как они составляют основу нашего понимания явлений в соответствующих сферах исследования и позволяют далее, отталкиваясь от точных результатов, строить приближенные теории для более сложных проблем.

К более сложным проблемам, связанным с задачей Кеплера, относятся, прежде всего, задачи небесной механики. Они послужили площадкой для разработки и развития различных методов аналитической механики, используя Солнечную систему в качестве экспериментальной лаборатории. При этом с самого начала исследований в этой области важную роль играло изучение численных данных о движении планет. С появлением компьютеров появилась возможность непосредственного численного моделирования динамики небесных тел. Кроме того, в недавнее время открылись широкие возможности визуализации различных явлений небесной механики, что имеет особую важность в данном контексте в связи с тем, что часто эти явления носят геометрический характер.

Таким образом, имеет смысл осуществить численное моделирование самой задачи Кеплера, чтобы, опираясь на имеющиеся для нее аналитические результаты, вникнуть в сферу вопросов, связанных с моделированием движений планет, и проиллюстрировать изученные выше закономерности.

Итак, мы имеем задачу определения движения тела в поле притягивающего центра[1)], задаваемом законом всемирного тяготения.

Для повышения точности расчетов применим при решении дифференциальных уравнений метод Адамса, дающий в большинстве случаев более высокую точность, чем использованный ранее метод трапеций.

3.7.2. Метод Адамса численного решения систем обыкновенных дифференциальных уравнений

Данный метод относится к классу многошажных, то есть использующих значения функций, задающих производные, и самих динамических переменных, вычисленные на нескольких предыдущих шагах. Вообще говоря, имя Адамса носит целая группа линейных многошажных методов определенного вида. Мы остановимся здесь на явном методе четвертого порядка (явные методы данной группы называют методами Адамса – Бэшфорта). Этот алгоритм для системы уравнений

$$\dot{\vec{x}} = \vec{F}(\vec{x}, t)$$

задается следующей явной (не содержащей искомые величины в правой части) формулой[2)]:

$$\vec{x}_{n+1} = \vec{x}_n + \frac{\Delta t}{24}(55\vec{F}_n - 59\vec{F}_{n-1} + 37\vec{F}_{n-2} - 9\vec{F}_{n-3}),$$

[1)] Под притягивающим центром подразумевается тело, принимаемое неподвижным. Такое приближение применимо, когда это тело имеет массу, много большую массы движущегося тела. Можно показать, однако, что к решению задачи для притягивающего центра сводится и общая задача двух тел с любыми массами.

[2)] Обоснование применимости этой формулы и последовательное исследование условий точности работы метода выходят за рамки настоящего изложения.

где $\vec{x}_k$ – как обычно, рассчитанные значения $\vec{x}$ для моментов $t_k = k\,\Delta t$, а $\vec{F}_k = \vec{F}(t_k)$ – значения правых частей решаемой системы в эти моменты. Как видно, в формуле используется значение $\vec{F}_n$, которое необходимо вычислить на данном шаге, а также три значения с предыдущих шагов, которые должны храниться в памяти. Таким образом, многошажность метода приводит к определенным затратам памяти, что, впрочем, не является критичным для рассматриваемого круга механических задач с небольшим числом степеней свободы, для которых эти затраты невелики[3)].

Метод имеет четвертый порядок точности. При этом, являясь явным, он не требует итерационной процедуры решения алгебраической системы, благодаря чему показывает бóльшую временную эффективность, чем неявные методы, в частности метод трапеций. Однако это же свойство явности приводит к неприменимости метода для решения жестких систем уравнений. Ввиду этого важно контролировать точность его работы путем сравнения с другими алгоритмами, контрольных просчетов с уменьшенным шагом, а также следя за точностью сохранения интегралов движения.

Поскольку метод использует значения функции $\vec{F}$ в три предыдущих момента, то возникает вопрос о том, как осуществить первые три шага интегрирования. Для этого можно использовать одношажные методы, например весьма популярный явный одношажный метод Рунге – Кутты четвертого порядка. Вычисления в нем проводятся по следующей схеме:

$$
\begin{aligned}
&\vec{k}_1 = \vec{F}(\vec{x}_n, t_n),\\
&\vec{k}_2 = \vec{F}(\vec{x}_n + \tfrac{1}{2}\Delta t\,\vec{k}_1, t_n + \tfrac{1}{2}\Delta t),\\
&\vec{k}_3 = \vec{F}(\vec{x}_n + \tfrac{1}{2}\Delta t\,\vec{k}_2, t_n + \tfrac{1}{2}\Delta t),\\
&\vec{k}_4 = \vec{F}(\vec{x}_n + \Delta t\,\vec{k}_3, t_n + \Delta t),\\
&\vec{x}_{n+1} = \vec{x}_n + \tfrac{\Delta t}{6}(\vec{k}_1 + 2\vec{k}_2 + 2\vec{k}_3 + \vec{k}_4).
\end{aligned}
$$

Таким образом, для вычислений по методу Адамса мы будем использовать вспомогательную стартовую процедуру, выполняющую три шага по методу Рунге – Кутты с сохранением значений функции $\vec{F}$ в узловых точках $(\vec{x}_n, t_n)$.

[3)] Этот фактор становится существенным при большом числе динамических переменных.

3.7.3. Программная реализация

Общая структура требуемой программы[1)] типична для задач, связанных с решением обыкновенных дифференциальных уравнений, и похожа на две предыдущие программы для задач о резонансе.

Остановимся на отличительных чертах данного случая. Во-первых, необходимо определить набор фазовых переменных, для которых мы собираемся рассчитывать эволюцию, задаваемую системой уравнений первого порядка. Вообще говоря, исходно в задаче имеются три пространственные координаты – и, соответственно, три скорости[1)] (то есть компоненты вектора скорости). Однако, поскольку выше было показано, что траектории в нашей задаче являются плоскими, то для исследования отдельной траектории можно сразу ограничиться плоским случаем[2)]. Итак, выбираем в качестве переменных $x, y, \dot{x}, \dot{y}$. Отсюда имеем структуру переменных задачи:

```
struct Var   // переменные системы первого порядка
{
    double x, y;
    double vx, vy;
};
```

Для нее, как обычно, определяются операции сложения и умножения на число, используемые в алгоритмах численного интегрирования. Далее необходимо определить параметры нашей задачи. Обратимся к уравнению динамики (3.2):

$$m\,\frac{d^2}{dt^2}\vec{r} = -\frac{\gamma\, m\, M}{r^3}\,\vec{r}.$$

Видно, что масса движущегося тела в него, по сути дела, не вхо-

[1)] Полный текст программы см. на стр. 93.

[1)] Терминология, называющая каждую производную от координаты скоростью, идет из аналитической механики, где часто используются системы координат, в которых нельзя отдельно выделить координаты каждого тела и соответственно исчезает понятие радиус-вектора и вектора скорости данного тела.

[2)] Такого рассмотрения недостаточно, если стоит цель изучить взаимное расположение орбит, лежащих в разных плоскостях, что часто требуется в небесной механике.

дит ("сокращается")[3]. При этом масса притягивающего центра и гравитационная постоянная входят только в виде произведения γM. Поэтому фактически мы имеем один параметр γM, который в программе назовем `GM`. Несмотря на то, что он оказался единственным, имеет смысл, тем не менее, сохранить введенную ранее структуру параметров `P`, так как это облегчит нам повторное использование уже написанных средств, привязанных к этой структуре, в частности процедуры метода трапеций. Кроме того, если потребуется впоследствии рассмотреть более сложную модель (например, с учетом дополнительных факторов), то весьма вероятно, что параметров станет больше. Итак, сохраняя единый стиль, задаем структуру параметров:

```
struct Par   // структура параметров задачи
{
    double GM;
    // коэффициент в потенциале, равный G * M
};
```

Записывая, как обычно, уравнение движения в виде системы первого порядка:

$$\begin{aligned}
\dot{x} &= v_x, \\
\dot{y} &= v_y, \\
\dot{v}_x &= -\frac{\gamma M x}{(x^2+y^2)^{\frac{3}{2}}}, \\
\dot{v}_y &= -\frac{\gamma M y}{(x^2+y^2)^{\frac{3}{2}}},
\end{aligned}$$

получаем функцию правых частей:

```
Var RHS(Var& V, Par& P, double t)
{   // правые части системы первого порядка
    Var Res;
```

[3] Это обстоятельство имеет глубокую природу. Его называют обычно эквивалентностью инертной массы (то есть той, которая задает связь ускорения тела с действующей на него силой по второму закону Ньютона) и гравитационной массы (то есть величины, определяющей силу гравитационного притяжения этого тела к другим телам по закону тяготения). Этот факт проверен с максимально доступной точностью, и есть веские доводы, что он не является случайным совпадением. С этим связаны идеи общей теории относительности, утверждающей, что гравитация есть проявление искривления телами пространства вокруг них.

```
    Res.x = V.vx;
    Res.y = V.vy;
    double r = sqrtl(V.x * V.x + V.y * V.y);
    double coeff = P.GM / (r * r * r);
    Res.vx = - coeff * V.x;
    Res.vy = - coeff * V.y;
    return Res;
}
```

Далее, реализуем алгоритмы численного интегрирования. Метод трапеций, ввиду единства обозначений, остается вообще без изменений. Добавляем метод Адамса – Бэшфорта 4-го порядка, о котором шла речь выше:

```
void Adams_Bashforth_4_step(Var& V, Var& F1, Var& F2,
                Var& F3, Par& P, double& t, double st)
{
    Var F = RHS(V, P, t);
    V = V + (st / 24.) * (  55. * F + (- 59.) * F1
                          + 37. * F2 + (- 9.) * F3 );
    t += st;
    F3 = F2;  F2 = F1;  F1 = F;
}
```

Здесь `F1, F2, F3` – правые части за 1, 2 и 3 шага до текущего момента соответственно, которые должны быть сохранены на предыдущих шагах и переданы в эту функцию. В ней самой вычисляются правые части `F` в текущий момент (до совершения шага), которые затем запоминаются в переменную `F1`, а остальные переменные "сдвигаются" на один шаг.

Первоначальное задание этих переменных происходит в стартовой процедуре, осуществляющей три шага по методу Рунге – Кутты:

```
void Start_for_Adams_4(Var& V, Var& F1, Var& F2,
            Var& F3, Par& P, double& t, double st)
{   // первые три шага,
    // с сохранением правых частей на каждом шаге
    F3 = Runge_Kutta_4_step(V, P, t, st);
    F2 = Runge_Kutta_4_step(V, P, t, st);
    F1 = Runge_Kutta_4_step(V, P, t, st);
}
```

Здесь предполагается, что функция `Runge_Kutta_4_step` должна возвращать структуру значений правых частей на момент начала шага, который в ней совершается (это имеет смысл, так как эти значения нужны в самой этой функции, поэтому можно не вычислять их за ее пределами повторно).

В соответствии с этим пишем сам метод Рунге – Кутты четвертого порядка:

```
Var Runge_Kutta_4_step(Var& V, Par& P, double& t,
                                              double st)
{   // возвращает правые части на момент начала шага
    double st2 = st / 2.;
    Var K1 = RHS(V, P, t);
    Var K2 = RHS(V + st2 * K1, P, t + st2);
    Var K3 = RHS(V + st2 * K2, P, t + st2);
    Var K4 = RHS(V + st * K3, P, t + st);
    V = V + (st / 6.) * (K1 + 2. * K2 + 2. * K3 + K4);
    t += st;
    return K1;
}
```

Перейдем к определению функции `main()`. Основой является стандартный основной цикл, в котором выполняется шаг алгоритма численного интегрирования. Далее, вывод вычисленных значений осуществим также привычным способом – будем выводить значения `x` и `y` в соответствующие файлы с интервалом времени `point_out_interval` (так как в графиках может оказаться достаточным более низкое разрешение, чем то, которое получается при отображении точек траектории на каждом шаге интегрирования; проведя несколько экспериментов с различными значениями `point_out_interval`, нужно будет выбрать максимальное, при котором получается кривая, на которой не видно изломов или отдельных точек). Аналогично поступаем с выводом на экран ошибок в интегралах движения (для контроля точности работы численного метода). Имеются два интеграла: энергия

$$E = \frac{mv^2}{2} - \frac{\gamma M m}{r}$$

и z-компонента углового момента:

$$L_z = m(x\, v_y - y\, v_x).$$

Нужно запомнить их значения в начальный момент времени и затем, с некоторым интервалом, выводить отклонения текущих значений от начальных.

Итак, полный текст программы выглядит следующим образом.

Заголовочный файл "Kepler.h":

```
#ifndef KEPLER_H
#define KEPLER_H

struct Var   // переменные системы первого порядка
{
    double x, y;
    double vx, vy;
};

// арифметические операции над структурами переменных:
Var operator +(Var& A, Var& B);
Var operator *(double k, Var& A);

struct Par   // структура параметров задачи
{
    double GM;
    // коэффициент в потенциале, равный G * M
};

// функция, вычисляющая правые части
// системы первого порядка:
Var RHS(Var& V, Par& P, double t);

// функции, вычисляющие интегралы движения:
double Energy(Var& V, Par& P);
double Momentum(Var& V);

// шаг метода трапеций:
void Trapezium_step(Var& V, Par& P, double& t,
                                        double st);

// шаг метода Рунге-Кутты четвертого порядка:
```

```
Var Runge_Kutta_4_step(Var& V, Par& P, double& t,
                                        double st);

// стартовая процедура для метода Адамса:
void Start_for_Adams_4(Var& V, Var& F1, Var& F2,
           Var& F3, Par& P, double& t, double st);

// шаг явного метода Адамса четвертого порядка:
void Adams_Bashforth_4_step(Var& V, Var& F1, Var& F2,
                Var& F3, Par& P, double& t, double st);

#endif
```

Файл алгебраических операций над структурами переменных "var.cpp":

```
#include "Kepler.h"

Var operator +(Var& A, Var& B)
{   // сложение двух структур переменных
    Var Res;
    Res.x = A.x + B.x;
    Res.y = A.y + B.y;
    Res.vx = A.vx + B.vx;
    Res.vy = A.vy + B.vy;
    return Res;
}

Var operator *(double k, Var& A)
{   // умножение структуры переменных на число
    Var Res;
    Res.x = k * A.x;
    Res.y = k * A.y;
    Res.vx = k * A.vx;
    Res.vy = k * A.vy;
    return Res;
}
```

Файл смысловых функций, связанных с задачей Кеплера "Kepler.cpp":

```
#include "Kepler.h"
#include <math.h>

Var RHS(Var& V, Par& P, double t)
{   // правые части системы первого порядка
    Var Res;
    Res.x = V.vx;
    Res.y = V.vy;
    double r = sqrtl(V.x * V.x + V.y * V.y);
    double coeff = P.GM / (r * r * r);
    Res.vx = - coeff * V.x;
    Res.vy = - coeff * V.y;
    return Res;
}

double Energy(Var& V, Par& P)
{   // интеграл энергии
    return 0.5 * (V.vx * V.vx + V.vy * V.vy)
        - P.GM / sqrtl(V.x * V.x + V.y * V.y);
}

double Momentum(Var& V)
{   // интеграл - z-компонента углового момента
    return V.x * V.vy - V.y * V.vx;
}
```

Файл методов численного интегрирования ОДУ "integ.cpp":

```
#include "Kepler.h"
#include "math.h"

void Trapezium_step(Var& V, Par& P, double& t,
                                        double st)
{
    Var V0 = V;   // структура переменных
                  // на предыдущем шаге
    Var F0 = RHS(V0, P, t);   // значение правых частей
                              // на предыдущем шаге
    Var V_prev = V;
```

```
    // вспомогательная структура, хранящая значение
    // вычисляемой новой структуры переменных,
    // полученное на предыдущей итерации

    do   // цикл решения алгебраического уравнения
    {     // методом простых итераций
        V_prev = V;

        V = V0 + 0.5 * st * ( F0
             + RHS(V_prev, P, t + st) );
    }
    while ( fabsl(V.x - V_prev.x)
          + fabsl(V.v - V_prev.v) > 0.00000001 );
  // сравнение результатов текущей и предыдущей
  // итераций в норме, равной сумме модулей компонент;
  // вычисление останавливается, если норма разности
  // меньше заданного значения

    t += st;   // сдвиг текущего времени на шаг
}

Var Runge_Kutta_4_step(Var& V, Par& P, double& t,
                                          double st)
{   // возвращает правые части на момент начала шага
    double st2 = st / 2.;
    Var K1 = RHS(V, P, t);
    Var K2 = RHS(V + st2 * K1, P, t + st2);
    Var K3 = RHS(V + st2 * K2, P, t + st2);
    Var K4 = RHS(V + st * K3, P, t + st);
    V = V + (st / 6.) * (K1 + 2. * K2 + 2. * K3 + K4);
    t += st;
    return K1;
}

void Start_for_Adams_4(Var& V, Var& F1, Var& F2,
            Var& F3, Par& P, double& t, double st)
{   // первые три шага,
    // с сохранением правых частей на каждом шаге
```

```
    F3 = Runge_Kutta_4_step(V, P, t, st);
    F2 = Runge_Kutta_4_step(V, P, t, st);
    F1 = Runge_Kutta_4_step(V, P, t, st);
}

void Adams_Bashforth_4_step(Var& V, Var& F1, Var& F2,
                Var& F3, Par& P, double& t, double st)
{
    Var F = RHS(V, P, t);
    V = V + (st / 24.) * (  55. * F + (- 59.) * F1
                          + 37. * F2 + (- 9.) * F3 );
    t += st;
    F3 = F2;  F2 = F1;  F1 = F;
}
```

Управляющий файл "main.cpp":

```
#include "Kepler.h"
#include <fstream>
#include <iostream>

using namespace std;

int main()
{
    Var V;
    V.x = 1.;   V.y = 0.;
    V.vx = 0.;   V.vy = 1.1;
    Par P;   P.GM = 1.;
    double t = 0., st = 0.001;
    double E0 = Energy(V, P);
    double L0 = Momentum(V);

    ofstream x_res_file("x.res");
    // файл значений x численной траектории

    ofstream y_res_file("y.res");
    // файл значений y численной траектории

    double t1 = 0;
```

```
double point_out_interval = 0.01;
// вспомогательные переменные
// для вывода в файлы значений координат
// с интервалом времени point_out_interval

double t2 = 0;
double error_interval = 1;
// вспомогательные переменные
// для вывода на экран ошибок в интегралах движения
// с интервалом времени error_interval

x_res_file << V.x << " ";
y_res_file << V.y << " ";

Var F1, F2, F3;
Start_for_Adams_4(V, F1, F2, F3, P, t, st);

while (t < 50)   // рассчитываем траекторию
{    // на интервале времени (0, 20)

    // шаг интегрирования:
    Adams_Bashforth_4_step(V, F1, F2, F3,
                                P, t, st);

    // другой вариант: Trapezium_step(V, P, t, st);

    t1 += st;
    if ( t1 >= point_out_interval )
    {   //вывод численного решения в файл
        x_res_file << V.x << " ";
        y_res_file << V.y << " ";
        t1 = 0;
    }

    t2 += st;
    if ( t2 >= error_interval )
    {   // вывод на экран
        // ошибок в интегралах движения
```

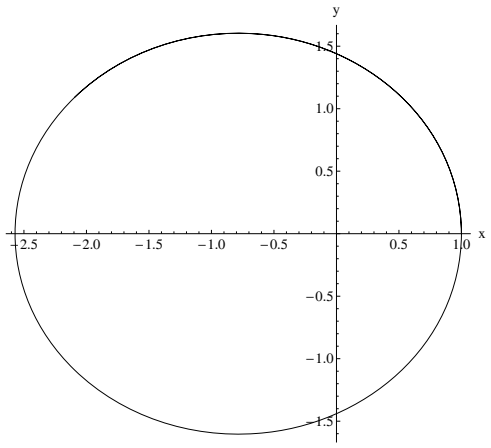


Рис. 3.1. Эллиптическая траектория, $\gamma = 1$, $M = 1$, $m = 1$, $\vec{r}_0 = (1, 0)$, $\vec{v}_0 = (0, 1.2)$.

```
            cout << "Time: " << t << "; ";
            cout << "Steps: "
                 << (int) (t / st) << "; \n";
            cout << "Error in E: "
                 << Energy(V, P) - E0 << ";     ";
            cout << "Error in L: "
                 << Momentum(V) - L0 << "\n\n";
            t2 = 0;
        }
    }

    x_res_file.close();
    y_res_file.close();

    return 0;
}
```

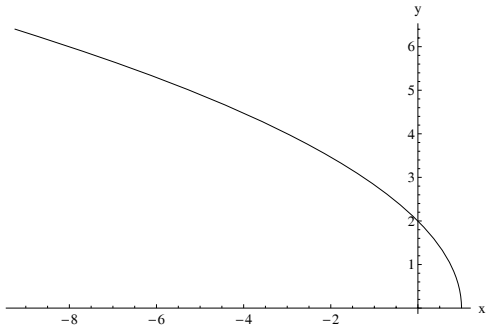


Рис. 3.2. Параболическая траектория, $\gamma = 1$, $M = 1$, $m = 1$, $\vec{r}_0 = (1, 0)$, $\vec{v}_0 = (0, \sqrt{2})$.

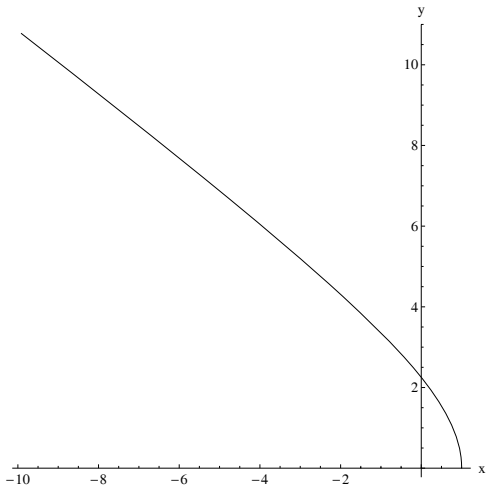


Рис. 3.3. Гиперболическая траектория, $\gamma = 1$, $M = 1$, $m = 1$, $\vec{r}_0 = (1, 0)$, $\vec{v}_0 = (0, 2)$.

3.7.4. Исследование задачи с помощью программы

Итак, зададим в нашей задаче некоторые начальные условия и осуществим расчет траектории. Выводимые на экран значения ошибок в интегралах позволяют контролировать точность работы численных методов и определять допустимую величину шага интегрирования. Так, в нашем случае для метода Адамса имеем при шаге 0.001 ошибки порядка 10^{-12}, что согласуется с тем, что метод имеет 4-й порядок, и дает подходящую точность.

Выведем траекторию в графическом виде (на плоскости). Как и предсказывает теория, здесь возможны три варианта: эллиптическая траектория (рис. 3.1), параболическая (рис. 3.2) и гиперболическая (рис. 3.3). Какой случай реализуется – зависит от знака энергии: соответственно $E < 0, E = 0$ или $E > 0$.

Далее, проиллюстрируем утверждение второго закона Кеплера: за равные промежутки времени радиус-вектор заметает равные площади. Для этого будем изображать радиус-вектор точки на траектории с некоторым интервалом времени. Результат (для начальных условий рисунка 3.1 эллиптической орбиты и с интервалом вывода радиус-вектора 1) показан на рис. 3.4. Видно, что чем ближе тело к притягивающему центру, тем быстрее оно движется: заметается "сектор" меньшего "радиуса" , но с бóльшим углом при вершине (так как $r^2\dot{\phi} = const$, то, в первом приближении, сектор вдвое меньшего радиуса, чем данный, будет иметь вчетверо больший угол, и тело будет пролетать вдвое большее расстояние в этом секторе).

Кроме того, мы можем наглядно пронаблюдать закономерность, выражаемую третьим законом Кеплера: квадраты периодов орбит относятся как кубы больших полуосей. Для этого получим траекторию другого размера, например изображенную на рис. 3.5. По рисункам, подсчитывая секторы, заметаемые за единицу времени, и оценивая по шкале большие полуоси, получаем, что периоды этой и предыдущей орбит относятся как, приблизительно, $9 : 15 = 3 : 5$, а большие полуоси – приблизительно как $2.5 : 3.5 = 5 : 7$. Имеем $(\frac{3}{5})^2 = 0.36, (\frac{5}{7})^3 \approx 0.364$, что согласуется с третьим законом Кеплера (учитывая приближенность используемых значений; разумеется, более точное измерение периодов и полуосей приводит к более точному равенству).

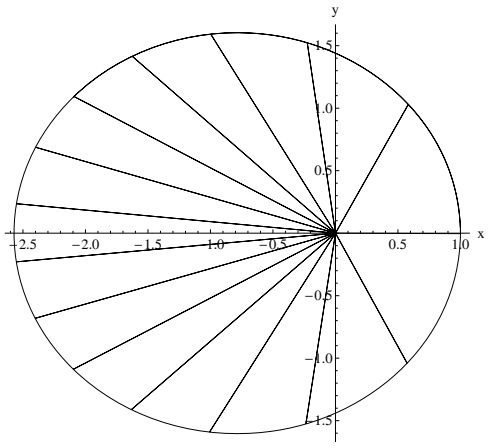


Рис. 3.4. Второй закон Кеплера: за равные промежутки времени заметаются равные площади.

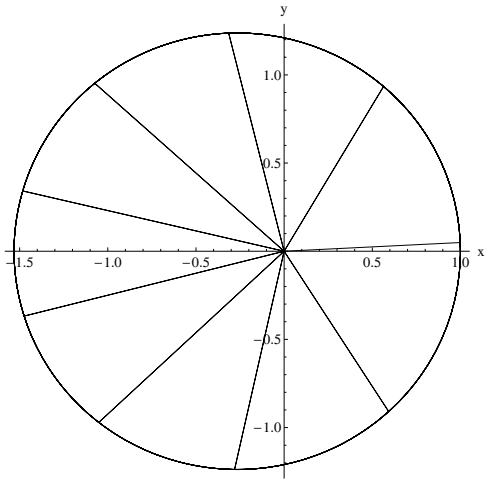


Рис. 3.5. К третьему закону Кеплера.

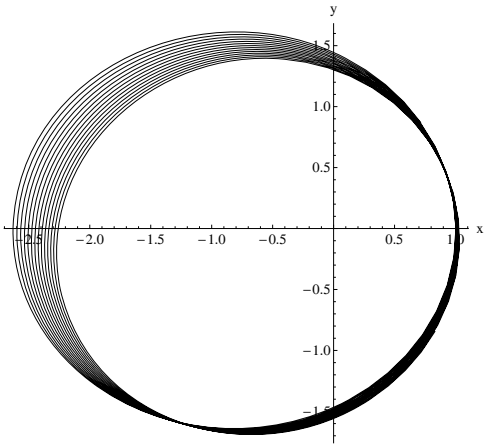


Рис. 3.6. Численные артефакты: траектория при шаге интегрирования 0.2.

3.7.5. Вопросы точности вычислений

Обсудим вопрос точности численного метода. В описанных вычислениях шаг интегрирования принимался равным 0.001. Это обеспечивало достаточную точность, что видно из сохранения интегралов движения (ошибки порядка 10^{-12}), а также из соответствия форм траекторий предсказаниям теории. Если же, однако, шаг недостаточно мал, например, для нашей системы, равен 0.2, то возникает траектория, изображенная на рис. 3.6. Видно, что траектория отличается от эллипса, и с каждым шагом отклонение растет. Ошибки в интегралах движения составляют порядка 10^{-2}.

Поучительно сравнить эту траекторию с другой ситуацией. Возьмем вместо использованного ранее потенциала тяготения, пропорционального $\frac{1}{r}$, потенциал с другим показателем, например, пропорциональный $\frac{1}{r^{1.01}}$. Результат показан на рис. 3.7. Видно, что происходит движение, напоминающее предыдущий случай. Траектория с каждым оборотом все больше отклоняется от первого витка. Вообще, можно показать, что для потенциала, отличного от $\frac{1}{r}$ (рассмотренная задача Кеплера) или r^2

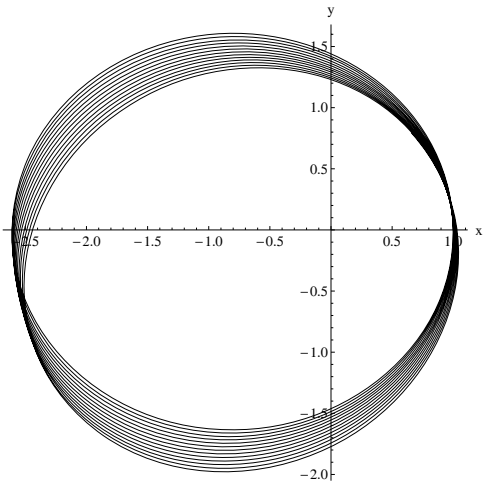


Рис. 3.7. Движение в потенциале, пропорциональном $\frac{1}{r^{1.01}}$.

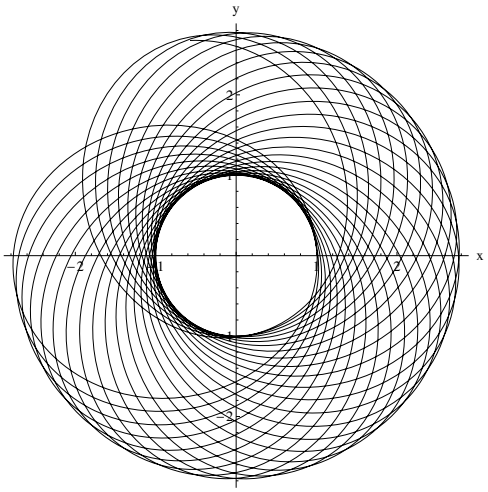


Рис. 3.8. Движение в потенциале, пропорциональном $\frac{1}{r^{1.05}}$.

(так называемый пространственный осциллятор), траектория представляет собой так называемую розетку (см. рис. 3.8).

Таким образом, мы видим, что нередко эффекты неточности численных методов могут принимать внешний вид, сходный с определенными свойствами дифференциальных уравнений, которые в действительности не имеют места в решаемой системе. При этом слишком большой размер шага является только одним из примеров, вовсе не исчерпывающим все факторы, которые могут привести к ошибкам. Поэтому при анализе результатов численных расчетов постоянно необходимо задаваться вопросом, чем на самом деле является та или иная особенность поведения компьютерной модели: проявлением свойства изучаемой системы или численным артефактом. Для решения этого вопроса можно использовать расчеты с другими параметрами численной схемы, другими методами, с повышенной точностью компьютерной арифметики, а также теоретические соображения, которые нередко могут предсказывать определенные следствия, сравнение которых с поведением численного решения позволяет распознать проявления ошибок расчетов.

3.7.6. Выводы

Итак, мы осуществили численное моделирование задачи о движении тела в поле ньютоновского притягивающего центра. При этом подтвердились три закона Кеплера: форма орбит, постоянство секториальной скорости и связь периода с большой полуосью. Ясно, что написанная программа может далее быть дополнена для моделирования более сложных задач небесной механики (например, задачи трех тел[1)]).

[1)] Задачи многих тел, да и сама задача Кеплера, могут приводить к численным трудностям в тех случаях, когда некоторые тела пролетают на малом расстоянии друг от друга: в эти интервалы времени на тела действуют очень большие силы, и точность работы метода интегрирования ухудшается. Для преодоления этого эффекта можно использовать алгоритмы с переменным шагом, в которых шаг уменьшается при больших величинах производных.

3.8. Возможности и проблемы численного моделирования

Появление компьютеров открыло качественно новые возможности для научных исследований. Следует заметить, что работа с численными данными проводилась задолго до этого события: Кеплер открыл свои законы благодаря численному анализу данных наблюдений Тихо Браге (Tycho Brahe) четыре столетия назад. И хотя единственным инструментом, по сути, было гусиное перо[2)] – недостаток технических средств с лихвой был восполнен собственными мозгами.

В настоящее время потенциал вычислительных средств по скорости выполнения арифметических операций достиг уровня, не сравнимого с возможностями человека, и с каждым годом этот потенциал увеличивается. Благодаря этому появилась возможность моделирования весьма сложных процессов, к которым до этого зачастую трудно было даже подступиться. Однако при всей вычислительной мощи, современные компьютеры не могут заменить воображение, фантазию, способность выдвигать гипотезы и делать обобщения – качества исследователя, необходимые для совершения больших открытий.

Кроме того, даже результаты, получаемые вычислениями на компьютерах, в большинстве случаев требуют активного участия человека: необходим контроль за правильностью вычислений[1)], интерпретация полученных результатов, разработка и улучшение численных методов, корректировка моделей.

Практика показывает, что надёжность и эффективность резко возрастают при одновременном применении аналитических методов и компьютеров.

[2)] Возможно, применялась также грифельная доска, но более для экономии: бумага стоила дорого.

[1)] Компьютер не ошибается, как человек, из-за усталости или невнимательности, однако имеет ряд собственных источников ошибок, таких, как конечноразрядная арифметика, необходимость дискретизации непрерывных задач (разностные схемы); и конечно, взаимодействие с компьютером человека нередко также вносит свою лепту.

Глава 4

Поверхности рода g > 0, реализованные как эквипотенциальные поверхности

Помимо все возрастающих вычислительных возможностей, компьютер дает также весьма удобные средства визуализации, отображения информации, в том числе и научных фактов и идей. Этот аспект приобретает все большее значение, так как технические элементы деятельности постепенно перекладываются на вычислительные машины, а идейные продвижения, по-прежнему, доступны практически только человеку. А значит, ключевую роль играют методы, позволяющие представлять информацию в виде, удобном для восприятия и анализа, дающем возможность выявлять в ней качественные закономерности. В связи с выделенной ролью зрительного восприятия и, соответственно, геометрического воображения, одним из наиболее продуктивных методов такого рода является графическое представление информации. Кроме того, объективная естественность применения графических изображений во многих задачах объясняется тем, что имеющиеся в них закономерности либо напрямую связаны с проявлениями геометрии нашего пространства (например, движения планет), либо, в более общем случае, эти закономерности могут проявляться в свойствах более абстрактных геометрических структур, связанных с этими задачами (например, пространства цветов).

В этой главе мы рассмотрим несколько примеров геометрических структур и вопросы их визуализации. При этом будем использовать систему Mathematica (приводимые далее инструкции написаны и исполнены в версии 7.0), предоставляющую богатые возможности графического отображения, а также удобные и эффективные средства набора и обработки формул. Начнем с кратких указаний по использованию самых основных графических средств.

4.1. Указания по использованию графических средств системы Mathematica

Работа в системе Mathematica происходит путем набора и исполения инструкций, записываемых в файл с расширением "nb" (notebook). Инструкции могут задавать значения переменным, давать команду вычислить введенное выражение или преобразовать его определенным образом, решить аналитически или численно аглебраическое или дифференциальное уравнение и т.д. Мы остановимся на командах построения графиков.

Для вывода графиков функций одной переменной имеется функция `Plot` (встроенные функции имеют имена, начинающиеся с большой буквы; в их названия, как правило, входят слова без сокращений; слова в одном названии записываются слитно, каждое слово с большой буквы; аргументы функций задаются в *квадратных* скобках, через запятую). Обязательными аргументами функции `Plot` являются функция, график которой требуется построить (аргумент пишется в квадратных скобках), и название аргумента с диапазоном его изменения, на котором нужно строить график, в виде `{x, a, b}` (x – аргумент, a – минимальное значение, b – максимальное). Например, инструкция

```
Plot[Sin[x],{x, 0, 10}]
```

выводит график функции синус для x, меняющегося от 0 до 10. После обязательных аргументов, через запятую, можно указать (в произвольном порядке) различные опции, настраивающие процедуру построения графика. Так, с помощью опции `PlotRange` можно указать минимальное и максимальное значения ординаты для прямоугольника, в котором будет изображаться график. Так, задание `PlotRange -> {0, 1}` приведет в нашем случае к выводу только верхней половины графика.

Для построения графиков по результатам численных расчетов требуется строить графики по наборам точек. Для этого имеется функция `ListPlot`. В нее передается список пар: значение агрумента – значение функции. Пары задаются в виде `{3, 4}`, список пар – в виде `{ {1, 5.2}, {2, 3.7}, {3, 4.1} }`. Списки значений аргумента и функции можно непосредственно считать из файлов.

Предположим, для простоты, что эти файлы лежат в той же директории, что и nb-файл, в котором мы строим графики. Тогда вначале необходимо установить эту директорию в качестве текущей:

```
SetDirectory[NotebookDirectory[]];
```

Далее, для считывания значений аргумента из файла `"x.res"` применяем команду

```
Lx = ReadList["x.res", Number,
              WordSeparators -> {","}];
```

Здесь `Number` – тип считываемых данных (числа), а инструкция `WordSeparators -> {","}` задает разделитель (в нашем примере числа в файле `"x.res"` разделены запятыми). В итоге в переменной `Lx` будет записан список значений аргумента вида `{ 0., 0.5, 1.0, 1.5, ... }`. Аналогично считываем список значений функции:

```
Ly = ReadList["y.res", Number,
              WordSeparators -> {","}];
```

Теперь можно создать список пар аргумент – функция, используя команду

```
Pairs = Table[{Lx[[i]], Ly[[i]]}, {i, 1, Length[Lx]}];
```

Здесь `Lx[[i]]` – i-й элемент списка `Lx`, функция `Table` создает список пар указанного вида `{Lx[[i]], Ly[[i]]}` для `i`, меняющегося от 1 до `Length[Lx]` – длины списка Lx.

Далее по этим точкам строим график с помощью команды

```
G = ListPlot[Pairs, Joined -> True]
```

Здесь опция `Joined -> True` указывает, что точки должны быть соединены отрезками. Получающийся график выводится на экран и запоминается в переменной `G`.

Далее, если необходимо, можно сохранить его в виде файла графического формата, например eps:

```
Export[NotebookDirectory[] <> "graph.eps", G,
          ImageSize -> 500]
```

Здесь первый аргумент функции `Export` – полное имя файла (включая путь), в который надо записать график. В нашем случае эта строка получена слиянием (оператор `<>`) пути к директории nb-файла со строкой – именем файла `"graph.eps"`. Опция `ImageSize -> 500` задает размер изображения в файле.

Указанным способом (с добавлением небольшого числа опций, настраивающих внешний вид изображения) были получены графики в этой книге.

Аналогично функциям `Plot` и `ListPlot`, строящим двумерные графики, для вывода трехмерных графиков функций от двух переменных существуют функции `Plot3D` и `ListPlot3D`. В первой из них надо задавать функцию двух переменных и диапазоны их изменения, во второй – список троек чисел: значения двух аргументов и значение функции. Эти функции строят поверхность графика функции на данном прямоугольнике изменения аргументов.

Применим их для иллюстрации фактов геометрии двумерных поверхностей.

4.2. Реализация поверхностей рода g > 0

Будем рассматривать двумерные поверхности в трехмерном евклидовом пространстве. Среди них можно выделить класс замкнутых поверхностей, то есть тех, у которых нет края: например, сфера является замкнутой поверхностью, а полусфера – нет. Кроме того, выделяется класс ориентируемых поверхностей. На качественном уровне это означает, что поверхность имеет две стороны, которые можно покрасить в разные цвета. В качестве примера можно привести, опять же, сферу (рис. 4.1), или цилиндр, или тор (рис. 4.2). Рассмотрим ориетнируемые замкнутые поверхности, причем не уходящие на бесконечность (как плоскость или бесконечно продолжаемый цилиндр).

Из числа названных выше к таковым относятся сфера и тор. Как видно между ними имеется существенное различие (которого нет, например, между сферой и эллипсоидом): у тора есть отверстие, а у сферы – нет. В более общем случае это свойство определяется следующим образом. Поверхности указанного вида можно продеформировать без разрывов и склеек так,

что они примут вид сферы с некоторым числом "приклеенных" к ней "ручек" (например, в случае одной ручки получаем поверхность тяжелоатлетической гири). Тогда упомянутым выше обобщенным свойством будет *род* поверхности – количество ручек в таком ее представлении. Таким образом, род сферы равен 0, род тора равен 1. Продолжая эту серию, естественно рассмотреть также поверхности рода 2, 3 и т. д. Возникает вопрос о том, как их реализовать, или, другими словами, как задать математически некоторые поверхности, которые имели бы данный род.

Для этого рассмотрим подробнее случай тора. Его можно себе представить как множество точек, расположенных на заданном расстоянии r от окружности радиуса R ($R > r$), расположенной, например, в плоскости XY и задаваемой уравнением: $x^2 + y^2 = R^2$. Это расстояние выражается формулой $d = \sqrt{(\sqrt{x^2+y^2} - R)^2 + z^2}$. Отсюда получаем поверхность тора как объединение графика функции $z = \sqrt{r^2 - (\sqrt{x^2+y^2} - R)^2}$ и его зеркального отражения в плоскости XY: $z = -\sqrt{r^2 - (\sqrt{x^2+y^2} - R)^2}$. Итак, тор можно получить в программе Mathematica следующей командой:

$R = 2;\ r = 0.5;\ l = R + r;$
Plot3D[{$\sqrt{r^2 - \left(\sqrt{x^2+y^2} - R\right)^2}$,
-$\sqrt{r^2 - \left(\sqrt{x^2+y^2} - R\right)^2}$},{x,-l,l},{y,-l,l},BoxRatios→ {1,1,0.2}]

Она изображает графики двух указанных функций на квадрате, задаваемом неравенствами $-l < x < l$, $-l < y < l$. Опция BoxRatios устанавливает относительные размеры (по трем осям) параллелепипеда, в котором будет находится график (размер по оси z берется меньше, чем по x и y, так как толщина тора меньше его диаметра).

Рассмотрим подробнее функцию $r^2 - (\sqrt{x^2+y^2} - R)^2$ (см. рис. 4.3). Она имеет максимум (равный r) в точках окружности $x^2 + y^2 = R^2$. При удалении от окружности она убывает, и на расстоянии, равном r, обращается в 0. На этом уровне мы

как бы отсекаем ее график, а взятие квадратного корня обеспечивает гладкость поверхности в точках линии пересечения ее с плоскостью XY. Как будет показано ниже, для наших целей было бы удобно иметь функцию с похожими свойствами, однако чтобы она стремилась по модулю к нулю при удалении от кольца. Действительно, предлагаемая идея построения поверхности рода 2 состоит в следующем. Такую поверхность можно представлять себе как два тора, слившихся в месте соприкосновения ("крендель"). Два слившихся тора можно получить указанным выше способом, если имеется функция, принимающая положительные значения в некоторой окрестности двух окружностей, расположенных близко друг к другу, например, задаваемых уравнениями: $(x+a)^2+y^2=R^2$ и $(x-a)^2+y^2=R^2$, где a больше R на некоторую небольшую величину. В качестве такой функции можно рассмотреть нечто подобное электростатическому потенциалу, который принимает большие по модулю значения вблизи заряженных тел и стремится к нулю при удалении от них. В силу этого свойства потенциалы разных тел, складываясь, дают функцию, принимающую большие по модулю значения вблизи любого из этих тел и стремящуюся к нулю при удалении от них. Однако потенциал в чистом виде нам не подходит в виду его ухода на бесконечность при приближении к заряженному телу. Подходящую функцию, в случае окружности, можно задать, например, выражением: $e^{\frac{-\left((x+a)^2+y^2-R^2\right)^2}{3}}$. Складывая такие функции для двух близких окружностей, получаем искомый результат. Таким образом, поверхность рода 2 получается следующей командой:

$R=2;\ a=2.3;\ l=5;\ m=3;$
Plot3D[{$(e^{\frac{-\left((x+a)^2+y^2-R^2\right)^2}{3}}+e^{\frac{-\left((x-a)^2+y^2-R^2\right)^2}{3}}-.1)^{\frac{1}{2}}$,
-$(e^{\frac{-\left((x+a)^2+y^2-R^2\right)^2}{3}}+e^{\frac{-\left((x-a)^2+y^2-R^2\right)^2}{3}}-.1)^{\frac{1}{2}}$},
{x,-l,l},{y,-m,m}, BoxRatios→ {1.5,1,0.2}]

Результат показан на рис. 4.4.

Аналогично получаем поверхность рода 3:

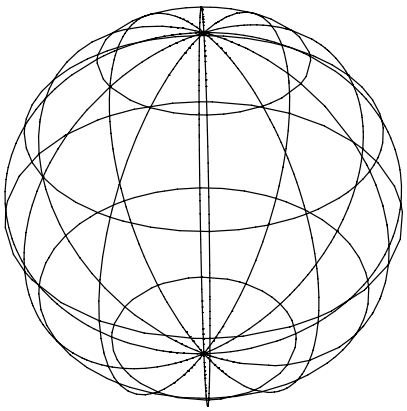


Рис. 4.1. Поверхность рода 0: сфера.

$R = 2;\ a = 2.2;\ b = 1.5;\ l = 5;\ m = 3;$
Plot3D[{($e^{\frac{-\left((x+a)^2+y^2-R^2\right)^2}{3}} + e^{\frac{-\left((x-a)^2+y^2-R^2\right)^2}{3}}$
$+e^{\frac{-\left(x^2+(y-a-b)^2-R^2\right)^2}{3}} - .1)^{\frac{1}{2}}$,-($e^{\frac{-\left((x+a)^2+y^2-R^2\right)^2}{3}}$
$+e^{\frac{-\left((x-a)^2+y^2-R^2\right)^2}{3}} + e^{\frac{-\left(x^2+(y-a-b)^2-R^2\right)^2}{3}} - .1)^{\frac{1}{2}}$},
{x,-l,l},{y,-m,2m},BoxRatios→ {1.2,1,0.3}]

Результат показан на рис. 4.5.

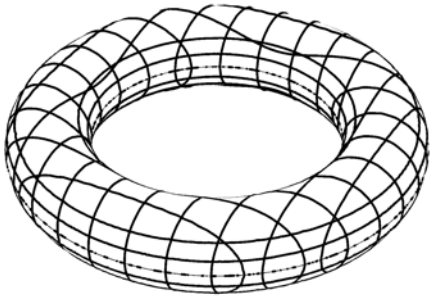


Рис. 4.2. Поверхность рода 1: тор.

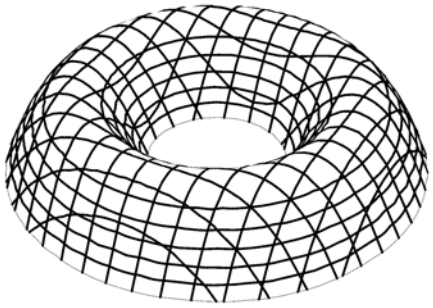


Рис. 4.3. Функция, используемая для задания тора (см. текст).

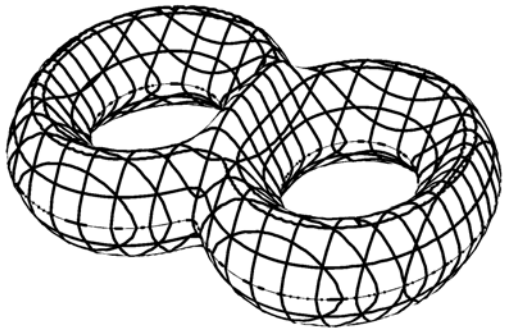


Рис. 4.4. Поверхность рода 2: крендель.

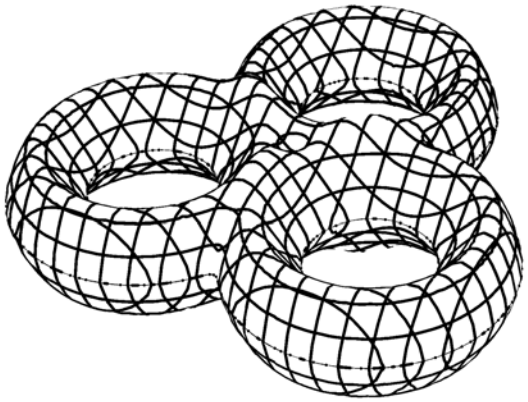


Рис. 4.5. Поверхность рода 3.

Голо Войслав Любомирович
Синицын Дмитрий Олегович

Компьютерное моделирование и визуализация задач механики и геометрии

М., Издательство Центра прикладных исследований при механико-математическом факультете МГУ, 2009. – 120 стр.

Подписано в печать 07.02.2009 г.
Формат 60×90 1/16. Объем 7,5 п.л.
Заказ 2 Тираж 300 экз.

Издательство ЦПИ при механико–математическом факультете МГУ
г. Москва, Воробьевы горы.

Отпечатано на типографском оборудовании механико-математического факультета