1. Решение уравнений движения для одной или нескольких частиц

Постановка задачи

Решение уравнений движения является классической задачей механики. Для произвольного набора частиц они представляются системой дифференциальных уравнений вида

$$m\frac{d^2\vec{r}_i}{dt^2} = \vec{F}_i\,, \qquad (1.1)$$

где $\vec{r}_i(t) = (x_i, y_i, z_i)$ – радиус вектор i-той частицы (i= 0,1,…N). Сила F_i является в общем случае функцией координат и скоростей, её явный вид определяется условиями задачи.

Для декартовых координат и скоростей одной частицы имеем

$dx/dt=v_x$,

$dv_x/dt=a_x=F_x/m$

и аналогично для остальных координат y и z.

При невозможности найти точное решение уравнений движения (1.1), для расчетов координат и скоростей в произвольный момент времени применяется один из методов приближенного (численного) решения обыкновенных дифференциальных уравнений (ОДУ).

Самый простой из них – это метод Эйлера. Он позволяет находить координаты и скорости в момент времени $t+\Delta t$, если известны параметры движения в момент времени t:

$$x(t+\Delta t) = x(t) + v_x(t)\cdot\Delta t\,, \qquad (1.2)$$

$$v_x(t+\Delta t) = v_x(\mathrm{t}) + a_x(t)\cdot\Delta t. \qquad (1.3)$$

Повторяя эти действия N раз, можно рассчитать отрезок траектории за время ($N\cdot\Delta t$). Соответственно, в программе строчки (1.2), (1.3) запишутся в виде:

```
x:=x+vx*dt
vx:=vx+ax*dt
```

Формулы имеют следующий смысл: x «новое» равно x «старое» плюс малое смещение на величину vx*dt. Переменная dt есть величина шага, которая должна быть достаточно малой, чтобы погрешность вычислений не превысила допустимую величину.

Модификацией метода Эйлера применительно к уравнениям движения является метод Эйлера-Кромера:

$$v_x(t+\Delta t) = v_x(t) + a_x(t)\cdot\Delta t, \quad (1.4)$$

$$x(t+\Delta t) = x(t) + v_x(t+\Delta t)\cdot\Delta t. \quad (1.5)$$

В некоторых случаях он дает более устойчивый результат, в частности, в задаче о колебаниях.

Если имеется программа для решения уравнений движения частицы под действием какой-либо определенной силы, то код этой программы легко может быть изменен для случая произвольных сил. Для этого следует переписать строчку, определяющую ускорение, а результирующую силу находить по обычному правилу суперпозиции.

Движение под действием постоянной силы в вязкой среде

Рассмотрим задачу о движении тела, на которое действуют сила тяжести, направленная вертикально вниз, и сила сопротивления воздуха, величина которой прямо пропорциональна скорости.

Направим ось Y вертикально вверх, а ось X так, чтобы вектор начальной скорости лежал в плоскости XY. В этом случае задача становится двумерной. Компоненты вектора ускорения можно записать в виде

$$a_x(t) = -\alpha\, v_x(t),$$

$$a_y(t) = -g - \alpha\, v_y(t),$$

где α – коэффициент сопротивления воздуха.

Численное решение уравнений движения (1.1) для данного примера можно реализовать с помощью встроенной функции решения ОДУ из подходящего пакета прикладных программ. Например, в системе MathCAD можно использовать функцию **odesolve,** дающую решение ОДУ методом Рунге-Кутта. Главное достоинство такого подхода состоит в том, что решаемые уравнения записываются в стандартном математическом виде.

Рассмотрим кратко структуру документа MathCAD для этого примера. Сначала необходимым параметрам задачи присваиваются численные значения. Затем, после ключевого слова **given**, записываются уравнения движения и начальные условия, при записи которых используется символ «=» (а не знак присваивания). Результат решения в виде двумерного вектора координат частицы появляется после вызова функции **odesolve**. Окончательно документ приобретает следующий вид:

v:=100 α :=0.05 g :=9.8 φ:=45·(π/180)

vx:=v·cos(φ) vy:= v·sin(φ)

Given

$$x''(t) = -\alpha \cdot x'(t) \qquad x(0)=0 \qquad x'(0)=vx$$
$$y''(t) = -g - \alpha \cdot y'(t) \qquad y(0)=0 \qquad y'(0)=vy$$

$$\begin{pmatrix} x \\ y \end{pmatrix} := \text{Odesolve}\left[\begin{pmatrix} x \\ y \end{pmatrix}, t, 15\right]$$

Результат вычислений по этому документу (при $0 \le t \le 15$) представлен на рис. 1.1.

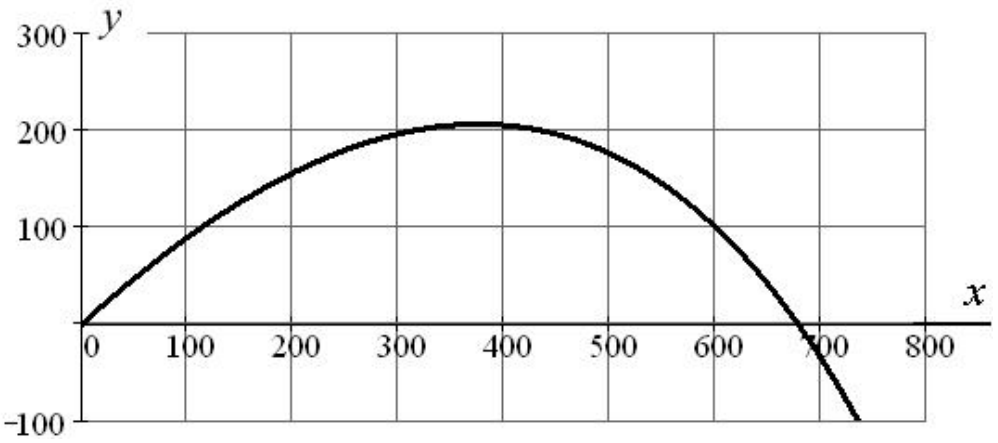

Рис. 1.1. Траектория частицы в поле силы тяжести
при учете силы сопротивления воздуха

Данную задачу можно также решить, написав программу на подходящем языке программирования высокого уровня, например, на языке Паскаль. Структура такой программы очевидна. Сначала после стандартного заголовка вводятся начальные данные: координаты частицы в момент времени t=0, угол бросания и модуль начальной скорости. Затем располагается блок численного решения ОДУ, где для краткости выбирается самый простой метод – метод Эйлера-Кромера.

Основой такого рода программы построения траектории является цикл по времени, в котором, с помощью метода Эйлера-Кромера (1.4), (1.5) пошагово находятся новые скорости и координаты частицы. Соответствующий фрагмент кода программы имеет вид:

```
While not keypressed do
begin
t:=t+dt;
ax:= –a*vx;       ay:= –g – a*vy;
vx:=vx+ax*dt;    vy:=vy+ay*dt;
x:=x+vx*dt;       y=:y+vy*dt;
xx:=round(x):
yy:=420 − round(y);
```

```
putpixel(xx,yy,12);
end;
```

Если вставить в этот цикл строчку `if y<=0 then break`, вычисление закончится, когда траектория достигнет поверхности земли (с точностью до половины шага траектории), при этом последнее значение координаты x будет являться дальностью полета.

Добавим в задачу граничные условия, например, это могут быть условия абсолютно упругого отражения от границы заданной формы.

Абсолютно упругое отражение точечной частицы от поверхности определяется известным правилом: «угол падения равен углу отражения» при сохранении модуля вектора скорости. Математическое выражение для формулы преобразования вектора скорости в этом случае легко записать через вектор нормали $\vec{N}$ к поверхности в точке падения (касания). Вектор скорости после удара определяется вычитанием из вектора скорости до удара его удвоенной проекции на единичный вектор нормали:

$$\vec{v}_2 = \vec{v}_1 - 2\vec{N}(\vec{N} \cdot \vec{v}_1). \qquad (1.6)$$

В этом выражении индекс «2» относится к скорости сразу после удара, а индекс «1» – к скорости непосредственно перед ударом. Модуль вектора нормали $\vec{N}$ выбран равным единице для упрощения записи формул. В программе соответствующие строки будут иметь вид (nx, ny – компоненты нормали)

```
nv:=nx*vx+ny*vy;
vx:=vx − 2*nx*nv;
vy:=vy − 2*ny*nv;
```

В рамках такого подхода рассмотрим задачу о построении траекторий для набора одинаковых невзаимодействующих частиц, падающих вертикально вниз на наклонную плоскость, на которой лежит шар (рис. 1.2.). Все отражения будем считать абсолютно упругими.

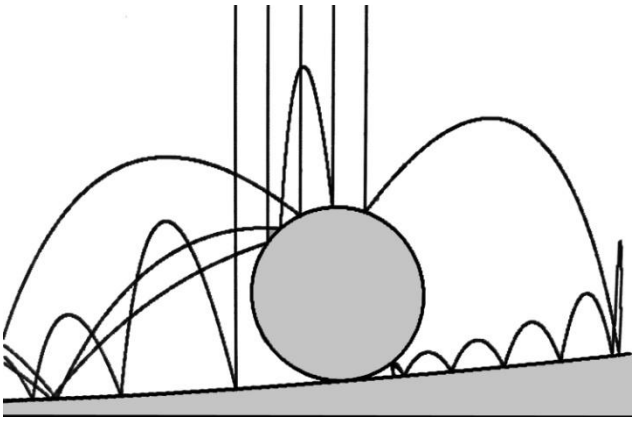

Рис. 1.2. Абсолютно упругое рассеяние потока частиц на шаре,
лежащем на наклонной плоскости.
На частицы действуют сила тяжести и сила сопротивления.

Код основной части программы для этой задачи имеет вид

```
1.  BEGIN
2.  Initgraph(gm,gd,'');
3.  fff:=0.12;
4.  nnx:=-sin(fff);
5.  nny:=cos(fff);
6.  line(1,420,round(600*nny),420-round(-nnx*600));
7.  r:=80;   x0:=325;  y0:=120;  dt:=0.01;
8.  xx:=round(x0); yy:=420-round(y0);
9.  circle(xx,yy,r);
10. J:=5;                          {число траекторий}
11. for i:=1 to J do begin
12. t:=0;  vx:=0; vy:=-40; x:=200+i*30; y:=400; a:=0.1;
13. while t<50 do
14. begin
15. t:=t+dt;
16. ax:=−a*vx; ay:= −g−a*vy;
17. vx:=vx+ax*dt; vy:=vy+ay*dt;
18. x:=x+vx*dt; y:=y+vy*dt;
19. xx:=round(x); yy:=420 – round(y);
20. putpixel(xx,yy,10);
21. if (y<( −nnx*x/nny)) then          {условие касания прямой}
22. begin
23. nv:=nnx*vx+nny*vy;
24. vx:=vx− 2*nnx*nv;
25. vy:=vy− 2*nny*nv;
26. y:=(-nnx*x/nny);
27. end;
28. if (sqr(x-x0)+sqr(y-y0)<sqr(r)) then  {условие касания круга}
```

```
29. begin
30. nx:=(x0-x)/r;
31. ny:=(y0-y)/r;
32. nv:=nx*vx+ny*vy;
33. vx:=vx-2*nx*nv;
34. vy:=vy-2*ny*nv;
35. x:=x+vx*dt; y:=y+vy*dt;
36. end;
37. end;
38. end;
39. readln; END.
```

Здесь основой программы также является цикл по времени (строки 13 – 37) для пошагового нахождения каждой i-той траектории методом Эйлера-Кромера, который дополнительно содержит блоки, определяющие изменение скорости при отражении от границ. Для отражения от плоскости это строки (21 – 27), для отражения от шара это строки (28 – 36).

Системы взаимодействующих частиц

При описании эволюции системы из N частиц, как правило, приходится решать соответствующую систему дифференциальных уравнений движения

$$m_i \frac{d\vec{r}_i}{dt} = \vec{F}_i = \vec{F}_i^{\,вн} + \sum_{\substack{j=1 \\ j\neq \mathrm{i}}}^{N} \vec{F}_{i,j}^{\,(\text{взаим.})}\,, \qquad (1.7)$$

где $i = 1\ldots N$. и введено разделение сил на внешнюю (по отношению к системе) силу и силы взаимодействия между частицами.

С математической точки зрения эта задача принципиально не отличается от задачи, требующей решения одного уравнения движения для отдельной частицы. Новое физическое содержание появляется при построении модели взаимодействия между частицами. Наиболее популярны следующие типы взаимодействия:

1. Абсолютно упругое отражение при столкновении частиц, рассматриваемых в приближении «твердых шаров». В этом исключительном случае уравнения движения решать не нужно. Задача сводится к преобразованию скоростей в момент столкновения.

2. Дальнодействующие центральные силы ($\vec{F}_{i,j}^{(\text{взаим})} = \vec{F}(\vec{r}_i - \vec{r}_j)$), например – взаимодействие кулоновского типа.

3. Квазиупругое взаимодействие (модель шариков и пружин). Число частиц в общем случае может быть произвольным, однако

реальные расчеты ограничиваются набором из нескольких сотен или тысяч частиц (этот случай рассматривается в разделе 6).

Кроме выбора типа взаимодействия следует также определить и граничные условия, которые существенно зависят от физических свойств рассматриваемой системы. При моделировании поведения жидкостей и газов используют взаимодействие первых двух типов. Например, для случая газа в замкнутом сосуде рассматриваемую систему частиц помещают в замкнутый объем («ящик») с непроницаемыми стенками. Задать граничные условия для этой модели означает задать характер взаимодействия частиц со стенками. Для идеального газа принимают, что частицы абсолютно упруго отражаются от стенок. Такое граничное условие автоматически обеспечивает сохранение энергии системы. При моделировании свойств твердого тела используется третья модель или другие модели с более реалистичным потенциалом межатомного взаимодействия.

Центральные силы. Кулоновское взаимодействие

Рассмотрим набор частиц, для которых силы взаимодействия являются кулоновскими, то есть для любой пары частиц вектора сил взаимодействия параллельны линии, соединяющей частицы, а их величина обратно пропорциональна квадрату расстояния между ними. При этом сами вектора сил, естественно, противонаправлены по закону «действия-противодействия». Результирующая сила, входящая в формулу (1.7), определяется по принципу суперпозиции:

$$\vec{F}_i = \sum_{j \neq i} A_{ij} \cdot \frac{\vec{r}_i - \vec{r}_j}{\left|\vec{r}_i - \vec{r}_j\right|^3}, \qquad (1.8)$$

причем $A_{ij}=A_{ji}$ по третьему закону Ньютона. В суммировании по индексу j исключен вклад для $j = i$. В задачах электростатики A_{ij} прямо пропорционально произведению зарядов частиц $q_i q_j$, а для гравитационного взаимодействия – произведению масс $m_i m_j$.

Расчет траекторий для набора частиц можно легко получить из соответствующей программы для одной частицы. Прежде всего, необходимо координаты и скорости частиц сделать элементами массивов. Кроме того, при вычислении ускорений потребуется вычислять результирующие силы по принципу суперпозиции. В результате цикл по времени примет вид

```
While not keypressed do
begin
t:=t+dt;
```

```
for i:=1 to N do begin    {расчет ускорений}
      ax[i]:=0;  ay[i]:=0;
      for j:=1 to N do begin
            if j<>i then begin
            rx:= x[i] − x[j];  ry:= y[i] − y[j];
            r:=sqrt(rx*rx+ry*ry);
            ax[i]:= ax[i] + A[i,j]*rx/(r*r*r);
            ay[i]:= ay[i] + A[i,j]*ry/(r*r*r);
            end;
      end;
end;
for i:=1 to N do begin
vx[i]:=vx[i]+ax[i]*dt;    vy[i]:=vy[i]+ay[i]*dt;
x[i]:=x[i]+vx[i]*dt;       y[i]=:y[i]+vy[i]*dt;
xx:=round(x[i]):
yy:=420 – round(y[i]);
putpixel(xx,yy,z[i]);
end; end;
```

Матрица коэффициентов A[i,j] задается заранее, причем для случая одинаковых частиц удобно поделить её на массу, чтобы сразу вычислять ускорения.

Абсолютно упругий центральный удар

Данный пример воспроизводит финитное движение трех одинаковых шаров, двигающихся вдоль одной прямой, движение шаров слева и справа ограничено стенками (см. рис. 1.3). Шары абсолютно упруго сталкиваются между собой и также отражаются от стенок.

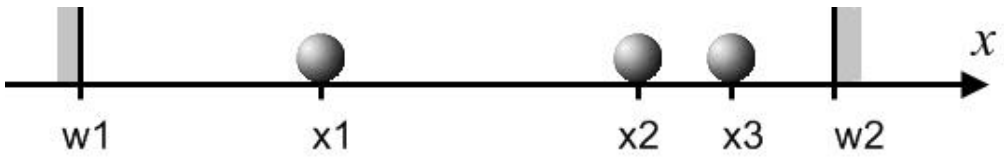

Рис. 1.3. Абсолютно упругое соударение трех шаров
в замкнутом одномерном пространстве

Формулы преобразования скоростей (известные из курса общей физики) удобно записать в симметричном по отношению к перестановке значков виде (штрих соответствует скоростям после удара):

$$v'_1 = -v_1 + 2V, \qquad v'_2 = -v_2 + 2V. \tag{1.9}$$

Здесь $V = \frac{m_1 v_1 + m_2 v_2}{m_1 + m_2}$ – скорость центра масс. Если массы совпадают, то происходит обмен скоростями: $v'_1 = v_2$, $v'_2 = v_1$.

Рассмотрим основной фрагмент программы, воспроизводящей движение трех шаров по указанному алгоритму. Результат представляется в виде зависимости координат шариков от времени.

```
1. While not keypressed do
2. begin
3. t:=t+dt;
4. x1:=x1+v1*dt;      x2:=x2+v2*dt;      x3:=x3+v3*dt;
5. if (x2−x1<=d) then
                          begin vvv:=v1; v1:=v2; v2:=vvv end;
6. if (x3−x2<=d) then
                          begin vvv:=v2; v2:=v3; v3:=vvv end;
7. if (x1<=w1) then v1:= −v1;
8. if (x3>=w2) then v3:= −v3;
9. xx1:=420 − round(x1):
10. xx2:=420 − round(x2):
11. xx3:=420 − round(x3):
12. putpixel(round(t),xx1,10);
13. putpixel(round(t),xx2,11);
14. putpixel(round(t),xx3,12);
15. end;
```

Расчет координат шариков x1, x2, x3 происходит согласно формулам x1:=x1+v1*dt; x2:=x2+v2*dt; x3:=x3+v3*dt; (строка 4).

Шарики движутся прямолинейно и равномерно, со скоростями v1, v2, v3 соответственно до момента соударения, в результате которого скорости соответствующих шариков скачком изменяются. Изменение скорости реализуется в соответствующем операторе if.

Рассмотрим, как происходит отражение шаров от стенок и соударение шаров между собой. При абсолютно упругом отражении от стенки в одномерном случае происходит мгновенное изменение знака скорости шарика, точнее через бесконечно малый интервал после момента касания шариком стенки, т.е. $v(t+0) = -v(t)$, если t – момент касания. Соответствующие строчки программы (7, 8) имеют вид:

```
if (x1<=w1) then v1:= −v1;
if (x3>=w2) then v3:= −v3;
```

Их всего две, так как со стенками взаимодействуют только два крайних шарика. При этом моменту соударения формально

соответствует строгое равенство координаты x шарика координате стенки ± радиус шарика. Однако, из-за приближенного характера вычисления координаты шарика, ее точное совпадение с предельным значением невозможно. Координата меняется мелкими скачками, и моментом соударения приближенно считается момент «первого пересечения», когда координата чуть-чуть пересекает запретную линию.

При абсолютно упругом соударении шариков одинаковой массы, они передают друг другу свои скорости (обмениваются скоростями):

$$v_1(t+0)= v_2(t)\ ;\quad v_2(t+0)= v_1(t) \qquad (1.10)$$

где t – момент времени, соответствующий касанию шариков. Касание шариков происходит в тот момент, когда расстояние между центрами шаров становится равным их диаметру d. С учетом приближенного характера вычислений, соударение описывают строки программы 5 и 6. Например: `if (x2−x1<=d) then begin vvv:=v1; v1:=v2; v2:=vvv end;` Промежуточная переменная vvv нужна для перемещения данных.

Таким образом, программа демонстрирует зависимость координат всех шариков от времени в виде единого графика (рис. 1.4).

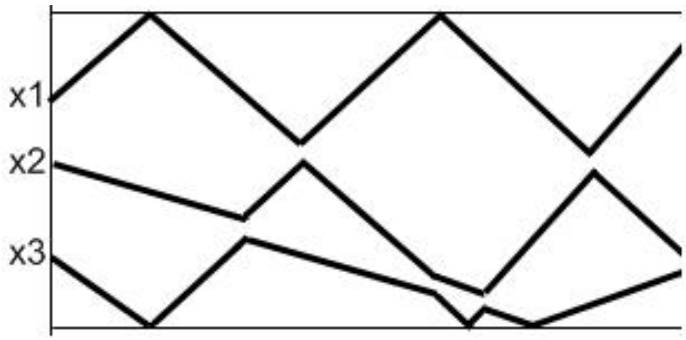

Рис. 1.4. Зависимость координат шаров от времени при центральном ударе. Минимальное расстояние между графиками равно диаметру шаров.

Абсолютно упругий нецентральный удар

Рассмотрим двумерную систему типа «шары в ящике» в рамках модели абсолютно твердых и идеально гладких шаров. Согласно курсу общей физики известно, что соударение происходит мгновенно, а изменение импульса при ударе происходит по линии, соединяющей центры шаров. Момент столкновения соответствует моменту соприкосновения, скорости шаров в результате столкновения меняются скачком по правилу: компоненты векторов скоростей шаров вдоль линии, соединяющей их центры, меняются, как при

центральном ударе, касательные к этому направлению компоненты не меняются.

В качестве примера выберем случай шаров одинаковой массы и диаметра *d*. Итак, пусть скорости первого и второго шаров до столкновения были $\vec{v}_1$ и $\vec{v}_2$ соответственно. Рассмотрим момент соприкосновения шаров (рис. 1.5). Введем единичный вектор $\vec{n}$ ($|\vec{n}|=1$), направленный по линии, соединяющей центры шаров, и перпендикулярный ему единичный вектор $\vec{L}$.

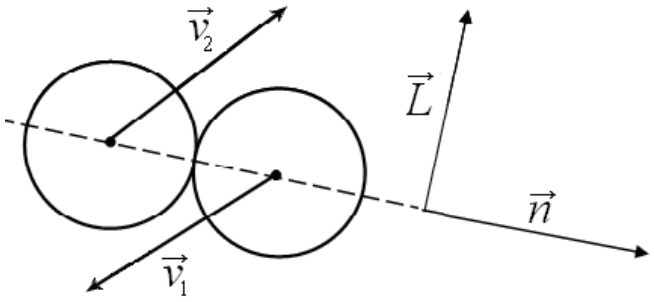

Рис. 1.5. Момент соприкосновения шаров при их соударении.

Если разложить вектора скоростей шаров до столкновения на взаимно перпендикулярные компоненты, соответствующие направлениям векторов $\vec{n}$ и $\vec{L}$, то очевидно получим

$$\vec{v}_1 = \vec{v}_{1\|} + \vec{v}_{1\perp} = (\vec{n}\cdot\vec{v}_1)\vec{n} + (\vec{L}\cdot\vec{v}_1)\vec{L}, \qquad (1.11)$$

$$\vec{v}_2 = \vec{v}_{2\|} + \vec{v}_{2\perp} = (\vec{n}\cdot\vec{v}_2)\vec{n} + (\vec{L}\cdot\vec{v}_2)\vec{L}, \qquad (1.12)$$

где значок $\|$ обозначает компоненту параллельную, а $\perp$ – перпендикулярную вектору $\vec{n}$.

Согласно приведенному выше правилу, компоненты скорости, перпендикулярные направлению $\vec{n}$ (т.е. $\vec{v}_{i\perp}$), при ударе не изменяются, в то время как компоненты $\vec{v}_{i\|}$ изменяются так же, как и при центральном ударе. Поскольку массы шаров одинаковы, они обмениваются продольными компонентами скоростей (см. (1.10)); то есть после столкновения $\vec{v}'_{1\|} = \vec{v}_{2\|}$ и $\vec{v}'_{2\|} = \vec{v}_{1\|}$, где скорости после соударения отмечены штрихом. Окончательно для скоростей после соударения получаем (сравни с (1.11), (1.12)):

$$\vec{v}'_1 = \vec{v}_{2\|} + \vec{v}_{1\perp} = (\vec{n}\cdot\vec{v}_2)\vec{n} + (\vec{L}\cdot\vec{v}_1)\vec{L}, \qquad (1.13)$$

$$\vec{v}'_2 = \vec{v}_{1\|} + \vec{v}_{2\perp} = (\vec{n}\cdot\vec{v}_1)\vec{n} + (\vec{L}\cdot\vec{v}_2)\vec{L}. \qquad (1.14)$$

Отметим, что результат не зависит от выбора знака векторов $\vec{n}$ и $\vec{L}$.

Выпишем теперь все входящие сюда формулы непосредственно в декартовых координатах. Пусть декартовы координаты центров шаров в момент соприкосновения есть (x_1, y_1) и (x_2, y_2). Тогда компоненты вектора $\vec{n}=(n_x, n_y)$ определяются формулами

$$n_x = (x_2 - x_1)/d\,, \quad n_y = (y_2 - y_1)/d\,, \tag{1.15}$$

где учтено, что в момент соприкосновения шаров $(x_1 - x_2)^2 + (y_1 - y_2)^2 = d^2$.

Удобно принять диаметр шаров за единицу измерения, т. е. все размеры задавать в единицах d. Тогда $n_x = x_2 - x_1$, $n_y = y_2 - y_1$, где (x_i, y_i) – координаты центров шаров, а условие соударения запишется в виде $n_x^2 + n_y^2 = 1$. Именно такой прием использован в программе.

Вектор $\vec{L}$ можно выбрать в виде $\vec{L} = (n_y, -n_x)$; очевидно, $\vec{L} \perp \vec{n}$, так как $(\vec{L} \cdot \vec{n}) = 0$, и $|\vec{L}| = 1$. Таким образом, с учетом (1.13), (1.14), для декартовых компонент скоростей после соударения получаем

$$v'_{1x} = n_x \cdot v_{2n} + n_y \cdot v_{1L}\,, \qquad v'_{1y} = n_y \cdot v_{2n} - n_x \cdot v_{1L}\,;$$

$$v'_{2x} = n_x \cdot v_{1n} + n_y \cdot v_{2L}\,, \qquad v'_{2y} = n_y \cdot v_{1n} - n_x \cdot v_{2L}\,;$$

где $v_{in} = (\vec{v}_i \cdot \vec{n}) = n_x\, v_{ix} + n_y\, v_{iy}$, а $v_{iL} = (\vec{v}_i \cdot \vec{L}) = n_y\, v_{ix} - n_x\, v_{iy}$.

В программе соударение шаров запишется в виде следующего блока:

```
nx=x2−x1; ny=y2−y1;    {компоненты вектора, соедин. центры шаров}
if (nx*nx+ny*ny<=1)   then              {условие касания шаров}
begin
if (ny>0) then ny=sqrt(1−nx*nx)}    {доп. нормировка вектора nx2+ny2=1}
          else ny= − sqrt(1−nx*nx);
vn1=nx*vx1+ny*vy1;      vL1=ny*vx1−nx*vум1;
vn2=nx*vx2+ny*vy2;      vL2=ny*vx2−nx*vy2;
vx1=nx*vn2 + ny*vL1;       vy1=ny*vn2−nx*vL1;
vx2=nx*vn1 + ny*vL2;       vy2=ny*vn1−nx*vL2;
end;
```

Данный блок следует вставить в цикл по времени, вычисляющий координаты шаров, который организован аналогично программе предыдущего раздела.