Майер Р.В., Глазовский пединститут

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ:
9. СИСТЕМЫ, СОСТОЯЩИЕ ИЗ БОЛЬШОГО ЧИСЛА ЧАСТИЦ

Существует целый класс **детерминированных методов** компьютерного моделирования твердых, жидких, газообразных тел, основанных на расчете энергии и силы взаимодействия между молекулами или частицами, составляющими эти тела, нахождения траекторий их движения, скоростей и координат. Альтернативой являются **стохастические методы** исследования различных состояний систем, состоящих из большого числа частиц. Они заключаются в многократном воспроизведении исследуемого процесса и статистической обработке получающихся результатов. Использование этих методов позволяет исследовать явления молекулярной физики, гидро- и газодинамики, физики твердого тела и т.д. [1-12].

9.1. Метод классической молекулярной динамики

Простейшей системой, состоящей из большого числа частиц, является газ. Ее **микросостояние** в данный момент времени характеризуется координатами и скоростями всех молекул, а **макросостояние** –- температурой, давлением и объемом (или температурой, энергией и объемом). Каждому макросостоянию соответствует большое число различных микросостояний. Результативным способом изучения этой системы на микроскопическом уровне является моделирование движения частиц с помощью компьютера. В этом и заключается **метод молекулярной динамики** (метод МД), предполагающий численное интегрирование уравнений движения и определение положения и скорости каждой частицы в моменты $\tau = t\Delta\tau$, где $t = 1,2,...$ Шаг по времени выбирают так, чтобы отклонение результата численного решения от точного с течением времени возрастало бы как можно медленней. Устойчивость вычислительной процедуры контролируется по выполнению закона сохранения энергии.

Вычислительные возможности ЭВМ не позволяют рассчитать движение всех 10^{23} молекул ($1/6$ моль), в вычислительных экспериментах моделируется система из $10^2 - 10^4$ частиц. Это соответствует очень небольшому объему газа, находящегося при нормальных условиях. Чтобы модель из 1000 частиц обладала свойствами реальной системы, сосуд разбивают на ячейки кубической формы с длиной ребра L и числом час-

тиц N. Ребра ячеек образуют трехмерную периодическую решетку. Средняя длина пробега и эффективный радиус сил межмолекулярного взаимодействия должны быть много меньше L. При этом можно считать, что трехмерный сосуд заполнен огромным числом одинаковых ячеек, молекулы в которых движутся похожим образом. Если молекула выходит из ячейки, то она входит в другую ячейку. Это означает, что с противоположной грани кубической ячейки внутрь должна войти такая же молекула с той же скоростью. Для того, чтобы представить движение всех молекул в сосуде, достаточно рассмотреть движение небольшого числа частиц в одной ячейке. Считается, что расположение молекул в других ячейках повторяется, они являются как бы копиями рассматриваемой ячейки (рис. 1.1), поэтому такие краевые условия называются **периодическими** [2, 7].

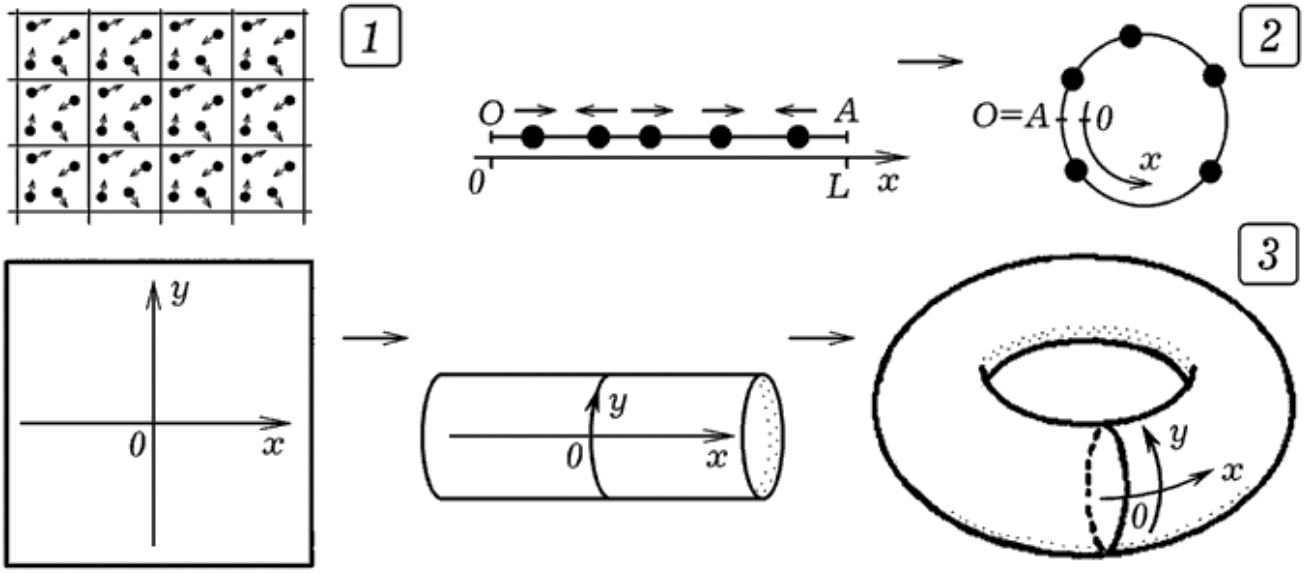

Рис. 1. Замыкание одно- и двумерной ячейки на саму себя.

В одномерной модели газа молекулы движутся вдоль отрезка OA длины L, отражаясь от его концов (стенок одномерного "сосуда"). Свернем отрезок в кольцо (рис. 1.2), замкнув его на самого себя, так, что частица, вышедшая из "сосуда" через правый конец A, заходит с той же скоростью через левый конец O. В двумерном случае квадратный "сосуд" превращается в тор (рис 1.3): вертикальные и горизонтальные стороны склеиваются так, что когда молекула выходит из сосуда через нижнюю стенку, такая же молекула с той же скоростью входит через верхнюю стенку.

При таком подходе число частиц в каждой ячейке остается неизменным, а их движение в различных ячейках согласованным, что не соответствует реальным системам и приводит к ошибкам. Для их уменьшения используют **стохастические граничные условия:** частица, пересекающая границу, удаляется из рассмотрения, но через случайные промежутки времени в случайные точки пространства в ячейку вводятся новые моле-

кулы имеющие случайные скорости. Это должно происходить так, чтобы сохранялись среднее число частиц (плотность), импульс системы и ее энергия. Рассмотренный выше подход используется для моделирования жидкостей и достаточно плотных газов [8].

Задание начальных координат и скоростей всех молекул определяется решаемой задачей. Если моделируется газ, частицы располагают беспорядочно так, чтобы они не накладывались друг на друга. При изучении жидкостей и твердых тел частицы располагают упорядоченно. Для исследования равновесных состояний начальные скорости должны быть заданы так, чтобы полная энергия соответствовала некоторой температуре и выполнялся закон сохранения импульса.

Сразу после запуска программы модель не находится в равновесном состоянии, так как частицы движутся с произвольными скоростями. Для того, чтобы проконтролировать достижение системой равновесного состояния, вычисляют среднюю энергию частицы, приходящуюся на каждую степень свободы, среднюю квадратичную флуктуацию скорости одной частицы, среднее расстояние между двумя частицами в ячейке. После релаксационного процесса перечисленные величины стремятся к равновесным значениям и флуктуируют вблизи них, а распределение частиц по скоростям соответствует закону Максвелла.

Согласно **теореме о равнораспределении энергии**, на каждую степень свободы молекулы приходится энергия $kT/2$, где T -- абсолютная температура. Если частицы имеют по s степеней свободы, то получаем:

$$\frac{s}{2}NkT = \sum_{i=1}^{N}\frac{m_i v_i^2}{2}, \text{ отсюда } T = \frac{1}{sNk}\sum_{i=1}^{N} m_i v_i^2 .$$

Давление равно отношению силы, с которой молекулы действуют на некоторую поверхность, к ее площади: $p = F/S$. Оно может быть найдено как плотность потока импульса через единичную площадку. В двумерном случае его находят так: мысленно разделяют прямоугольный сосуд горизонтальной линией длиной S и подсчитывают число молекул n, которые пересекли ее за время $\Delta\tau$. Тогда давление равно $p = m\upsilon/(S\Delta\tau)$.

Другой метод расчета давления состоит в использовании **теоремы вириала**. При этом давление связывают с изменением импульсов сил в системе и используют формулу, которая выведена в [2, с. 179-180]:

$$p = \frac{NkT}{V} + \frac{1}{sV}\left\langle \sum_{i=1}^{N} \vec{r}_i \vec{F}_i \right\rangle,$$

где $\vec{r}_i$ - радиус вектор i-ой молекулы, $\vec{F}_i$ -- равнодействующая всех сил, действующих на i-ую частицу со стороны всех других $N-1$ молекул, $s=2$ -- размерность модели.

Хаотическое движение молекул может быть охарактеризовано средним квадратом их смещения $R(\tau')^2 = <|\vec{r}_i(\tau_2)-\vec{r}_i(\tau_1)|^2>$, где $\tau=\tau_2-\tau_1$. Другой характеристикой движения частиц является **автокорреляционная характеристика скорости** $Z(\tau') = <\vec{v}_i(\tau_1)\vec{v}_i(\tau_2)>$, показывающая насколько их скорость в момент τ_2 тесно связана со скоростью в момент $\tau_1=\tau_2-\tau'$. У плотных газов и жидкостей **средняя длина пробега** молекул мала, поэтому уже при небольших разностях $\tau'=\tau_2-\tau_1$ величина $Z(\tau')$ стремится к нулю. В двумерном случае $Z(\tau') = v_{ix}(\tau_1)v_{ix}(\tau_2)+v_{iy}(\tau_1)v_{iy}(\tau_2)$ [2, 7]. Как следует из **эргодической гипотезы**, средние по времени значения физических величин, характеризующих данную молекулу, равны средним статистическим значениям по всем молекулам в данный момент τ.

9.2. Моделирование молекул твердыми сферами

Одна из простых моделей состоит в замене молекул твердыми сферами или шарами диаметром d, которые при соударении отталкиваются. Энергия взаимодействия, частиц центры которых удалены на r, равна:

$$U(r)=\begin{cases}\infty, & если \quad r\le d,\\ 0, & если \quad r>d.\end{cases}$$

В двумерном случае говорят от **твердых дисках**, а в одномерном -- о **твердых стержнях**. В рамках этих допущений математическая схема расчетов существенно упрощается, так как она не требует интегрирования системы уравнений Ньютона. Один из вариантов решения задачи сводится к следующему. На каждом шаге находят моменты времени между столкновениями молекул и определяют из этого ряда ближайший момент взаимодействия какой-либо пары молекул. После этого все частицы сдвигаются вдоль своих траекторий на соответствующие их скоростям расстояния, а скорости провзаимодействовавших молекул пересчитываются. После этого процедура расчета снова повторяется.

Возможен другой подход, требующий несколько больших затрат машинного времени. Координаты молекул при движении между соударениями рассчитываются через промежутки $\Delta\tau$, как $x_i^t = x_i^t + v_{ix}^{t+1}\Delta\tau$. Когда

i-ая и j-ая молекулы взаимодействуют (расстояние r_{ij} между их центрами меньше d), вычисляются их скорости после взаимодействия [3]. Можно приближенно считать, что если молекулы имеют одинаковые массы, то при ударе они просто обмениваются скоростями. При этом скорости молекул после упругого удара равны: $\vec{u}_i = \vec{v}_j$, $\vec{u}_j = \vec{v}_i$, где $\vec{v}_i$, $\vec{v}_j$ -- скорости молекул до удара. Изменив скорости, частицы следует отодвинуть друг от друга на расстояние, несколько большее d.

Покажем, что это не совсем правильно. Если j-ая частица покоится, то после удара с i-ой частицей, движущейся со скоростью $\vec{v}_{i0}$, получается, что она должна приобрести скорость $\vec{v}_j = \vec{v}_{i0}$, а i-ая частица -- остановиться ($\vec{v}_i = 0$). На самом деле так происходит только при центральном ударе. Если удар нецентральный, то частицы разлетаются так, что угол между $\vec{v}_i$ и $\vec{v}_j$ составляет $\pi/2$. Преимущество этого подхода в том, что он прост и обеспечивает сохранение энергии и импульса системы.

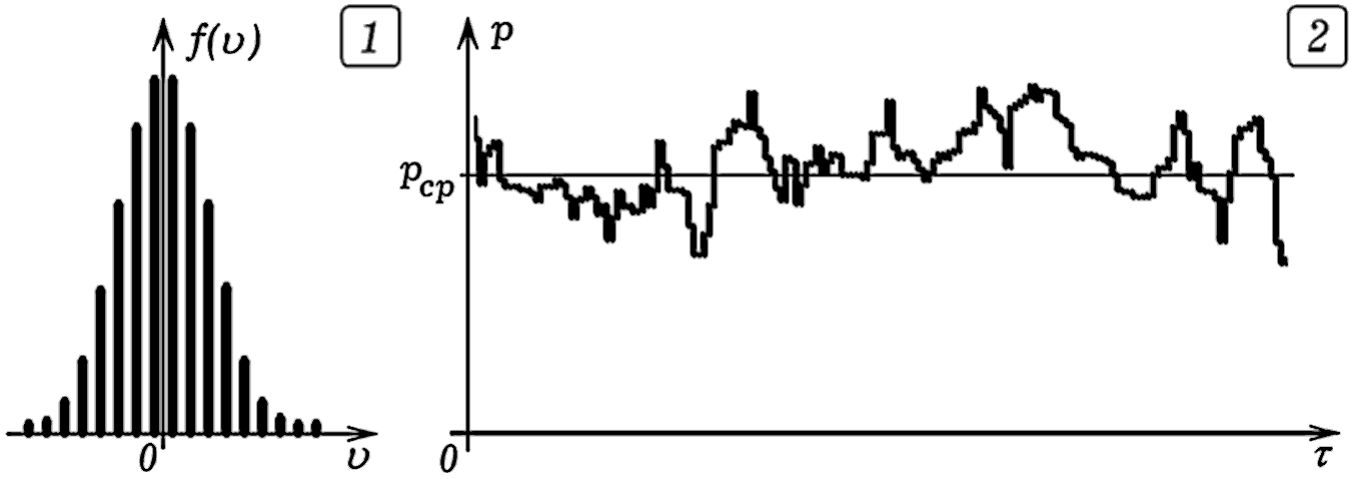

Рис. 2. Распределение молекул по скоростям. Давление газа.

Для моделирования молекул газа твердыми стержнями и твердыми дисками используются программы ПР-1 и ПР-2. В них заданы периодические граничные условия, вычисляются давление газа и температура. Можно предусмотреть отражение молекул от стенок сосуда: при упругом соударении со стенкой составляющая скорости молекулы, перпендикулярная к стенке, меняет свое направление на противоположное. На рис. 2.1 приведено получающееся распределение молекул газа по скоростям, соответствующее нормальному закону Гаусса. Из графика зависимости давления от времени (рис. 2.2) видно, что "мгновенные" значения давления p флуктуируют около своего среднего значения p_{cp}. После небольших изменений программы позволяют промоделировать движение молекул в поле тяжести, определить среднюю длину пробега молекул, сред-

ний квадрат смещения $R(\tau')^2$, автокорреляционную характеристику скорости $Z(\tau')$, изучить зависимость давления газа от температуры.

Используя модель "твердые сферы", ученым удалось исследовать **фазовый переход жидкость--твердое тело** и установить, что с ростом плотности системы коэффициент сжимаемости совершает резкий скачок [8]. Факт существования этого фазового перехода в системе твердых сфер означает, что его причина вызвана особенностями упаковки частиц, то есть обусловлена геометрией, а не силами межмолекулярного взаимодействия.

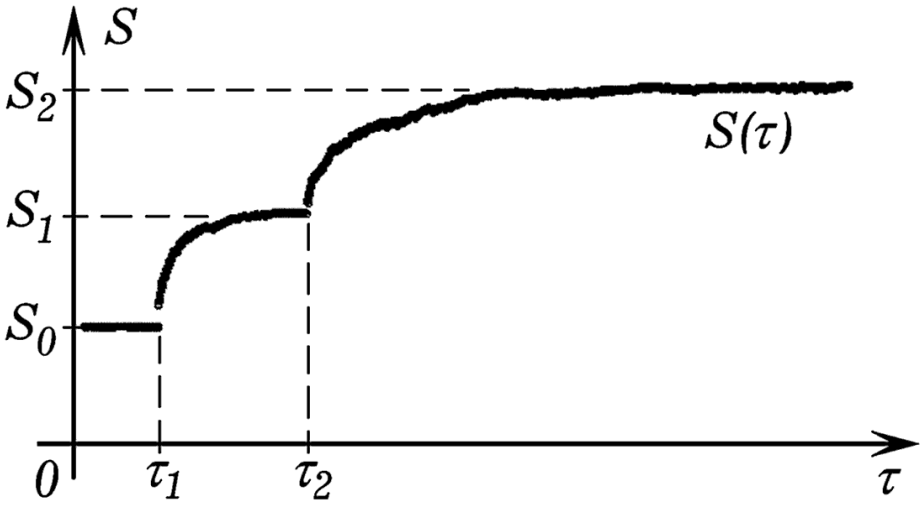

Рис. 3. Возрастание энтропии при диффузии газа.

Чтобы изучить увеличение энтропии при диффузии, можно использовать совсем простую модель, в которой движение молекул рассчитывается с помощью генератора случайных чисел. Представим длинный сосуд, разделенный двумя перегородкой на три части, которые открываются одна за другой в моменты τ_1 и τ_2. Сосуд полностью заполнен молекулами вещества 1, а в левой части еще имеются N молекул вещества 2. Разобьем сосуд на r равных элементарных объемов, число молекул вещества 2 в j-ом объеме обозначим n_j. Тогда энтропия системы равна:

$$S = -\sum_{j=1}^{r} p_j \ln p_j = -\sum_{j=1}^{r} \frac{n_j}{N} \ln \frac{n_j}{N}.$$

Используя программу Пр-4, можно промоделировать хаотическое движение молекул вещества 2, найти энтропию газа в последовательные моменты времени и получить график $S(\tau)$ в случае, когда открывается сначала одна, а потом другая перегородка (рис. 3). Диффузия -- необратимый процесс, в ходе которого энтропия возрастает.

9.3. Модель "взаимодействующие частицы"

Рассмотренная модель "твердые сферы" не учитывает притяжение молекул и поэтому не позволяет промоделировать объединение частиц в кластеры, конденсацию и кристаллизацию. Для исследования этих явлений используется модель Леннарда-Джонса, согласно которой потенциальная энергия и сила межмолекулярного взаимодействия равны [2, 8]:

$$U(r) = E\left(\left(\frac{\sigma}{r}\right)^{12} - 2\left(\frac{\sigma}{r}\right)^{6}\right), \quad F(r) = \frac{12E}{\sigma}\left(\left(\frac{\sigma}{r}\right)^{13} - \left(\frac{\sigma}{r}\right)^{7}\right).$$

Первое слагаемое отвечает за отталкивание частиц при малых r, второе -- за притяжение при больших r. Если $r = \sigma$, то силы отталкивания и притяжения компенсируют друг друга.

Рассмотрим систему из N частиц с массами m_i, $i = 1,2,...N$, находящуюся в силовом поле. Исходя из начальных координат x_{i0}, y_{i0} и скоростей $\vec{\upsilon}_{i0}$, необходимо рассчитать движение частиц в последующие моменты времени. Моделирование сводится к решению задача Коши для системы диффуравнений, получающихся из второго закона Ньютона:

$$\frac{d\vec{\upsilon}_i}{d\tau} = \vec{a}_i = \frac{1}{m}\left(\vec{F}_i + \sum_{j=1}^{N} \vec{F}'_{i,j}\right), \quad \frac{d\vec{r}_i}{d\tau} = \vec{\upsilon}_i, \quad i = 1,2,...N,$$

где $\vec{F}_i$ -- равнодействующая внешних сил, действующих на i-ую материальную точку со стороны тел, не входящих в систему, $\vec{F}'_{i,j}$ -- внутренняя сила, действующая на i-ую точку со стороны j-ой точки. Для этого организуют цикл по времени t, определяют проекции $R_{ix} = F_{ix} + F'_{ix}$, $R_{iy} = F_{iy} + F'_{iy}$ равнодействующей всех внешних и внутренних сил, действующих на каждую i-ую материальную точку в момент $t+1$, и записывают их в массивы. После этого переобозначают координаты всех частиц, записывая их в массивы xx[i], yy[i]. В цикле перебирают все молекулы и определяют проекции ускорения, скорости и координаты для каждой из них в момент $t+1$ по формулам:

$$a_{ix}^{t+1} = R_{ix}/m_i, \quad \upsilon_{ix}^{t+1} = \upsilon_{ix}^{t} + a_{ix}^{t+1}\Delta\tau, \quad x_i^{t+1} = x_i^t + \upsilon_{ix}^{t+1}\Delta\tau,$$
$$a_{iy}^{t+1} = R_{iy}/m_i, \quad \upsilon_{iy}^{t+1} = \upsilon_{iy}^{t} + a_{iy}^{t+1}\Delta\tau, \quad y_i^{t+1} = y_i^t + \upsilon_{iy}^{t+1}\Delta\tau.$$

Результаты сохраняют в массивах x[i], y[i], vx[i], vy[i]. Стирают изображения частиц в предыдущий момент времени t (массивы xx[i], yy[i]) и рисуют новые изображения, соответствующие моменту $t+1$. При необхо-

димости выводят значения температуры, давления и энергии в числовом виде. После этого все повторяется снова. Применяется алгоритм АЛ-1.

Алгоритм АЛ-1.

```
ПРОЦЕДУРА Sila {==
  ДЛЯ i:=1 ДО N ДЕЛАТЬ {A= Fx[i]:=0; Fy[i]:=0; =A}
  ДЛЯ i:=1 ДО N ДЕЛАТЬ {B= ДЛЯ j:=1 ДО N ДЕЛАТЬ {C=
    ЕСЛИ i<>j ТО {D= l:=sqrt(sqr(x[i]-x[j])+sqr(y[i]-y[j]));
      ЕСЛИ l<1 ТО l:=1;
      F:=-K1/sqr(l)+K2/sqr(l*l);
      Fx[i]:=Fx[i]+F*(x[i]-x[j])/l;
      Fy[i]:=Fy[i]+F*(y[i]-y[j])/l+m[i]*10; =D} =C} =B} ==}
НАЧАЛО ПРОГРАММЫ  N:=50; dt:=0.001;
  ДЛЯ i:=1 ДО N ДЕЛАТЬ {E=
    ЕСЛИ i>N/2 ТО m[i]:=2 ИНАЧЕ m[i]:=1;
    x[i]:=random(280)+60; y[i]:=random(280)+60;
    vx[i]:=random(30)-15; vy[i]:=random(30)-15;  E=}
  ПОВТОРЯТЬ ДО НАЖАТИЯ КЛАВИШИ {F=
    ВЫЗОВ ПРОЦЕДУРЫ Sila;
    ДЛЯ i:=1 ДО N ДЕЛАТЬ {G= xx[i]:=x[i]; yy[i]:=y[i];
      ax:=Fx[i]/m[i];          ay:=Fy[i]/m[i];
      vx[i]:=vx[i]+ax*dt;     vy[i]:=vy[i]+ay*dt;
      x[i]:=x[i]+vx[i]*dt;    y[i]:=y[i]+vy[i]*dt;
      ЕСЛИ x[i]<50 ТО x[i]:=349; ЕСЛИ y[i]<50 ТО y[i]:=349;
      ЕСЛИ x[i]>350 ТО x[i]:=51; ЕСЛИ y[i]>350 ТО y[i]:=51;
      СТЕРЕТЬ ТОЧКУ (xx[i], yy[i]);
      ПОСТАВИТЬ ТОЧКУ (x[i], y[i]); =G} =F}
КОНЕЦ ПРОГРАММЫ
```

Чтобы уменьшить погрешность вычислений и обеспечить сохранение энергии системы, используют **алгоритм Верле в скоростной форме**:
$x^{t+1} = x^t + \upsilon^t \Delta\tau + a^t(\Delta\tau)^2/2$, $a^t = F(x^{t+1})/m$, $\upsilon^{t+1} = \upsilon^t + (a^t + a^{t+1})\Delta\tau/2$.
Существует и другая математическая схема:

$$\frac{x_i^{t-1} - 2x_i^t + x_i^{t+1}}{\Delta\tau^2} = \frac{\vec{F}_{ix}^t}{m_i},\quad x_i^{t+1} = 2x_i^t - x_i^{t-1} + \frac{F_{ix}^t}{m_i}\Delta\tau^2,\quad \upsilon_{ix}^t = \frac{x_i^{t+1} - x_i^{t-1}}{2\Delta\tau}.$$

Также применяют **алгоритм с полушагом**: 1) вычисляют новые значения координат частицы: $x^{t+1} = x^t + \upsilon_x^t\Delta\tau + a_x^t\Delta\tau^2/2$; 2) рассчитывают промежуточное значение скорости $\upsilon_x^{t+1/2} = \upsilon_x^t + a_x^t\Delta\tau/2$; 3) определяют новое значение ускорения: $a_x^{t+1} = F_x^{t+1}(x^{t+1})/m$; 4) используя $\upsilon_x^{t+1/2}$ и a^{t+1}, находят новое значение скорости: $\upsilon_x^{t+1} = \upsilon_x^{t+1/2} + a_x^{t+1}\Delta\tau/2$.

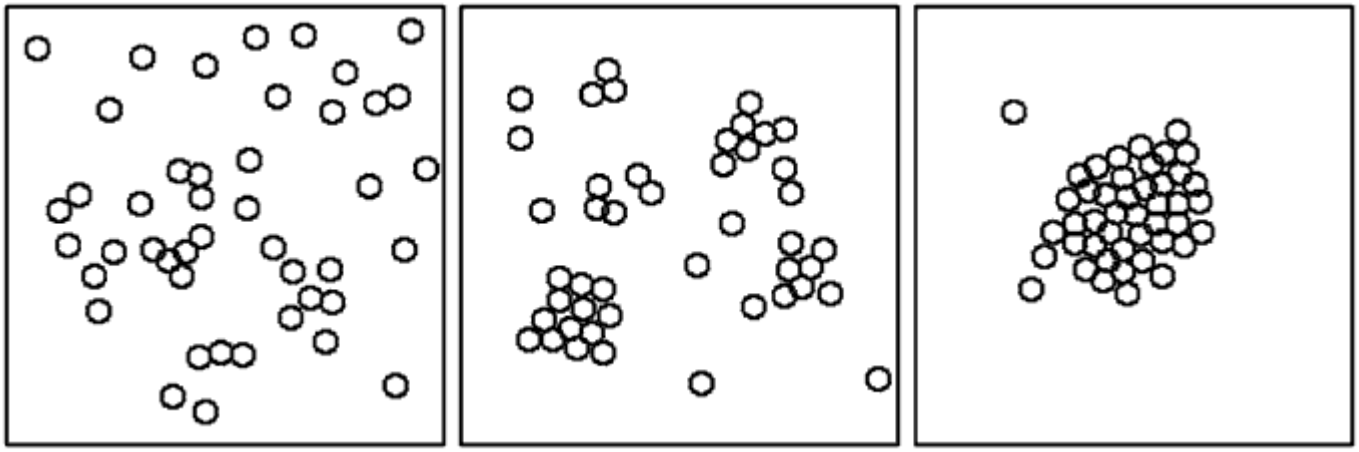

Рис. 4. Моделирование конденсации газа при охлаждении.

Этот подход реализован в программе ПР-5. С ее помощью можно промоделировать движение броуновской частицы, диффузию, конденсацию и испарение, рассчитать температуру, давление (обоими способами), средний квадрат смещения молекул $R(\tau')^2$, автокорреляционную характеристику скорости $Z(\tau')$. Для того, чтобы температура модели газа оставалась постоянной и равной T_1, используют условный оператор: **`If T>T1 then b:=0.999 else b:=1.001;`**. Коэффициент b входит в выражения для проекций скорости $\upsilon_{ix}^{t+1} = b\upsilon_{ix}^{t} + a_{ix}^{t+1}\Delta\tau$ и $\upsilon_{iy}^{t+1} = b\upsilon_{iy}^{t} + a_{iy}^{t+1}\Delta\tau$. Это позволяет "замедлять" и "разгонять" молекулы, то есть понижать и повышать температуру. При охлаждении происходит "конденсация": молекулы собираются в капли (рис. 4). При "нагревании" капли "испаряются", превращаются в газ.

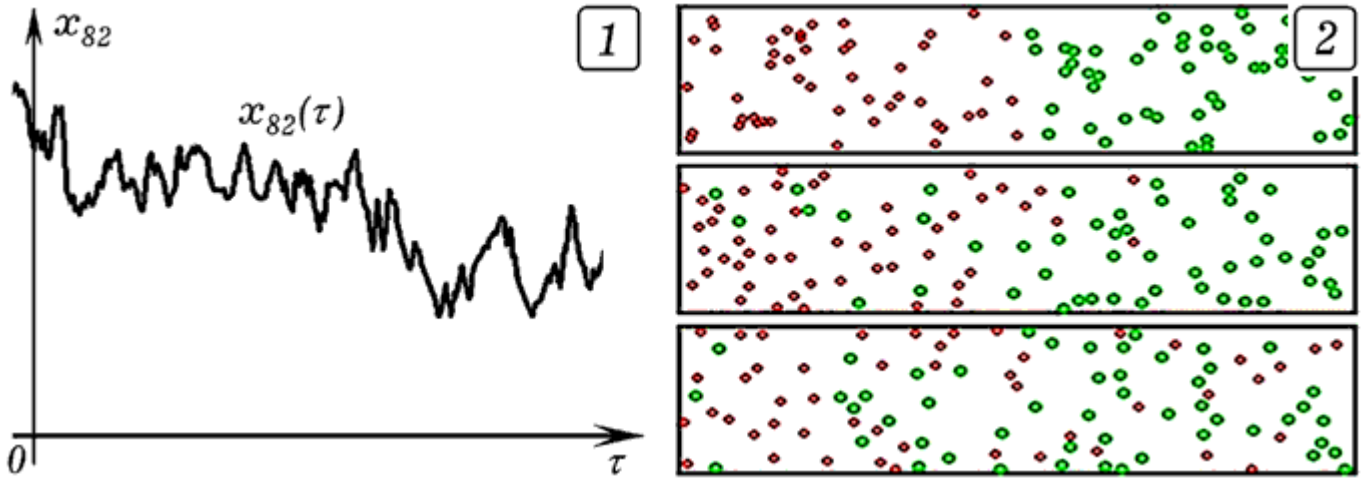

Рис. 5. Движение одной молекулы. Модель диффузии двух газов.

Для "наблюдения" случайного блуждания частиц и моделирования диффузии двух газов используется программа ПР-6, в которой учитывается только отталкивание молекул. Как видно из рис. 5.1, одна из коорди-

нат молекулы, находящейся вдали от стенки сосуда, изменяется случайным образом. Изначально "красные" молекулы с массами m_1 заполняют левую половину сосуда, а "зеленые" молекулы с массами m_2 -- правую (рис. 5.2). Вследствие хаотического движения молекул происходит их самопроизвольное перемешивание; количество частиц в обоих половинках сосуда выравнивается. Если массу одной из частиц сделать в 100 раз больше, а ее изображение выделить другим цветом, то удастся промоделировать броуновское движение. Внеся небольшие изменения в программу, можно получить на экране траекторию движения одной молекулы или броуновской частицы.

9.4. Развитие метода молекулярной динамики

Метод МД позволяет исследовать газ, жидкость, плазму, газовзвеси, суспензии, эмульсии, функционирование химических реакторов, доменных печей, двигателей внутреннего сгорания, ракетных двигателей. С его помощью удалось проверить разнообразные теории жидкости, уравнения состояния жидкостей и неидеальных газов, решать различные задачи теории переноса, изучить равновесные и неравновесные свойства среды, конденсацию, кристаллизацию, гидродинамические течения.

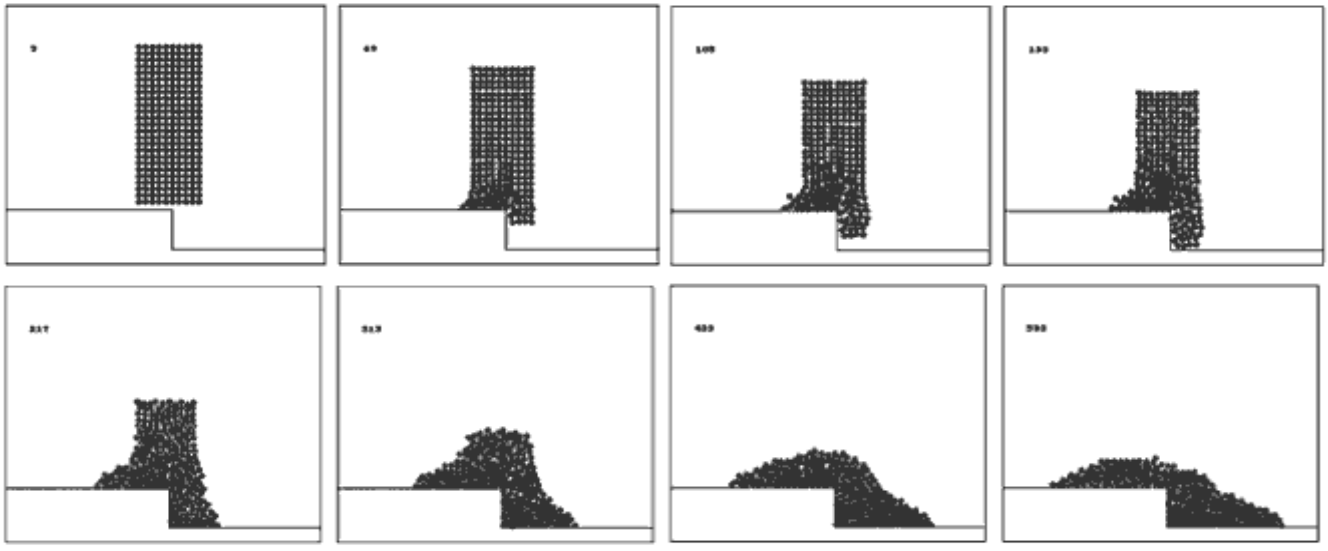

Рис. 6. Падение столба вязкой жидкости на выступ.

Развитием метода МД стал **метод частиц**, который заключается в представлении тела совокупностью крупных шарообразных частиц, взаимодействующих и движущихся в соответствии с законами классической механики. При удалении частицы притягиваются, а при сближении от-

талкиваются. Движение каждой частицы может быть рассчитано из второго закона Ньютона. Иногда используют **частицы-маркеры**, позволяющие визуализировать движение различных слоев жидкости или газа. В качестве примера использования метода крупных частиц рассмотрим модель падения вязкого тела на выступ. Компьютерная программа представлена в книге [5], результат моделирования -- на рис. 6. Закон взаимодействия макрочастиц подобран так, что тело теряет свою первоначальную форму, но не распадается на отдельные части.

Еще одним примером является обтекание газом препятствия (рис. 7.1). Используется программа из [5]. Частицы перемещаются вниз под действием внешней силы; достигая нижней границы расчетной области, они исчезают и появляются у ее верхней границы. Хорошо видно, что у передней стенки тела концентрация частиц (в значит и плотность газа) существенно выше средней, а за препятствием -- ниже. Аналогично можно промоделировать движение толпы людей и обтекание ей препятствия.

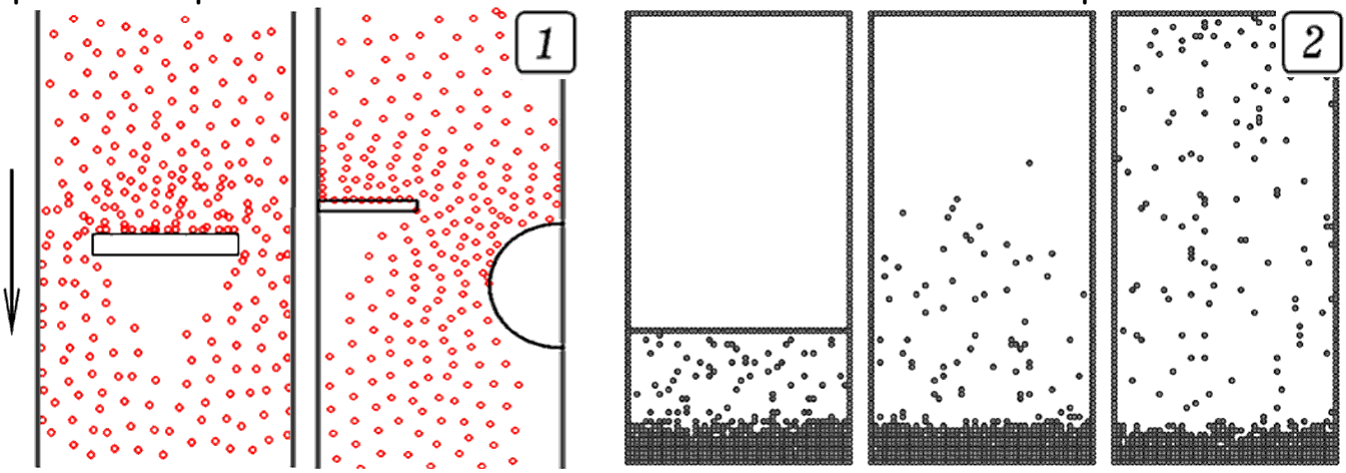

Рис. 7. Обтекание газом препятствия. Образование насыщенного пара.

Движение газа и жидкости может быть исследовано методом клеточных автоматов (КА). В главе 4 описан детерминированный КА, учитывающий столкновение частиц друг с другом. Рассмотрим вероятностный КА, позволяющий моделировать хаотическое движение молекул газа, испарение жидкости, образование насыщенного пара. Представим двумерный сосуд, разбитый на квадратные ячейки; в некоторых ячейках находятся молекулы, а остальные пусты. На каждом временном шаге ЭВМ случайно выбирает молекулу и с заданными вероятностями p_L, p_R, p_U, p_D, смещает ее в левую, правую, верхнюю или нижнюю ячейку, если она пуста. Для того, чтобы промоделировать расширение или диффузию газа, необходимо задать $p_L = p_R = p_U = p_D = 0{,}25$. КА будет моделировать насыщенный пар в случае, когда вероятности смещения молекулы

вверх (испарение, заполнение сосуда) и вниз (конденсация, переход в жидкость) зависят от концентрации молекул K в окрестности рассматриваемой молекулы с координатами (i, j). Для определения концентрации K будем подсчитывать число молекул в квадрате 5×5, построенном вокруг частицы (i, j). Если концентрация велика ($K>2$), то молекула с большей вероятностью идет вниз: $p_D=0{,}35$, $p_U=0{,}15$. Если концентрация молекул пара мала ($K\le2$), то молекула с большей вероятностью поднимается вверх: $p_D=0{,}15$, $p_U=0{,}35$. Смещения влево и вправо по прежнему равновероятны: $p_L=p_R=0{,}25$.

Используется программа ПР-7, результаты ее работы -- на рис. 7.2. Сначала моделируется испарение жидкости и образование насыщенного пара в небольшом объеме. Через некоторое время, когда наступает динамическое равновесие между паром и жидкостью, горизонтальная перегородка удаляется, и молекулы заполняют весь объем большого сосуда. Часть жидкости переходит в пар, который через некоторое время становится насыщенным.

9.5. Стохастическое моделирование молекул газа

Для получения результатов термодинамического характера не нужно точное знание координат и скоростей движения частиц системы в различные моменты времени. В этом случае используют статистические методы моделирования и вместо одной системы рассматривают **статистический ансамбль**, то есть совокупность большого числа одинаковых систем, находящихся в различных микросостояниях, которые соответствуют данному макросостоянию. Проанализируем несколько примеров.

Рассмотрим известную задачу о случайных блужданиях, впервые предложенную Пирсоном в 1906 г. Необходимо определить смещение пешехода, сделавшего N шагов равной длины в случайных направлениях. Ограничимся одномерным случаем и промоделируем движение молекулы газа, которая из-за хаотических соударений с другими частицами случайным образом перемещается вдоль оси Oz. При этом ее координата через равные промежутки времени $\Delta\tau$ принимает значения z^1, z^2, z^3, …, z^N, где N -- число шагов. Необходимо получить распределение конечной координаты z^N блуждающей точки при различном числе шагов N.

Будем использовать **метод статистических испытаний**. Программа ПР-8, содержит цикл, в котором случайным образом определяется направление и величина смещения молекулы на следующем временном шаге. Цикл совершает требуемое число итераций (равное количеству шагов N), в результате чего определяется конечная координата молекулы z^N. Для генерации случайных величин, имеющих равномерное распределение, применяется метод Фибоначчи с запаздываниями. Он состоит в использовании следующей рекуррентной формулы:

$$x_i = \begin{cases} x_{i-a} - x_{i-b}, & x_{i-a} \geq x_{i-b}, \\ x_{i-a} - x_{i-b} + 1, & x_{i-a} < x_{i-b}. \end{cases}$$

Здесь x_i -- действительные числа из диапазона $[0;\ 1]$, а $a = 55$, $b = 24$. Чтобы получить случайные числа из интервала $[-0{,}5;\ 0{,}5]$, из x_i вычитают $0{,}5$; их и прибавляют к z^t. Это позволяет осуществить 1000 реализаций N случайных смещений молекулы из начала координат, вычислить дисперсию конечной координаты z^N и изучить ее распределение.

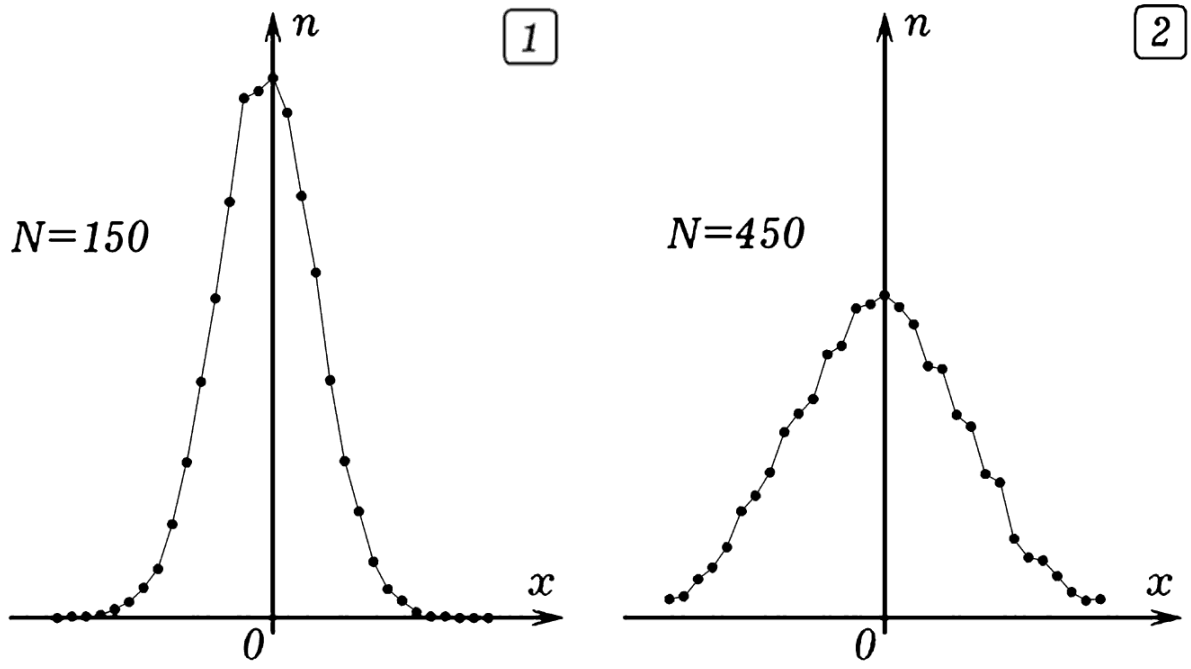

Рис. 8. Распределение z^N (10^4 испытаний, 150 и 450 шагов).

На рис. 8 показаны получающиеся распределения конечной координаты молекулы z^N после 150 и 450 шагов полученные в результате 10000 испытаний. Это нормальное распределение Гаусса [2, 7]. Можно отключить графический режим и вывести на экран средний квадрат дисперсии (СКО) z^N при заданном N. Если построить график зависимости среднего квадрата смещения "блуждающей" частицы $S^2 = <(z^N)^2>$ от

количества шагов N, то получится прямая пропорциональность (рис. 9). В данном случае моделируется **одномерная диффузия** молекул газа, которые сначала находились в небольшом объеме. Результаты можно интерпретировать так: 1) с течением времени увеличивается область, в которой может находится хаотически движущаяся молекула; 2) если из небольшого объема выходит 10000 молекул некоторого вещества, то через некоторое время (пропорциональное количеству шагов N) молекулы вследствие диффузии расползутся и займут большую область.

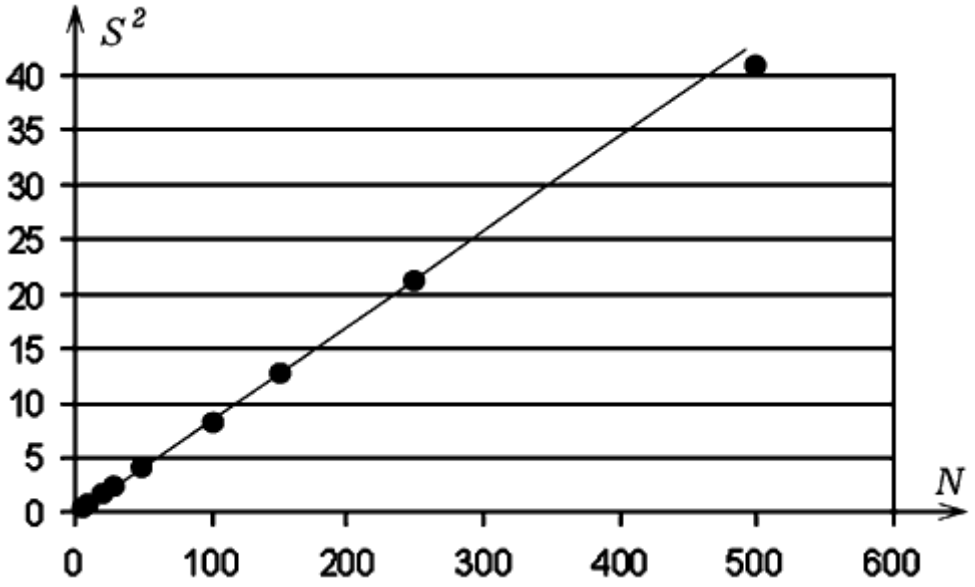

Рис. 9. Средний квадрат смещения z^N пропорционален N.

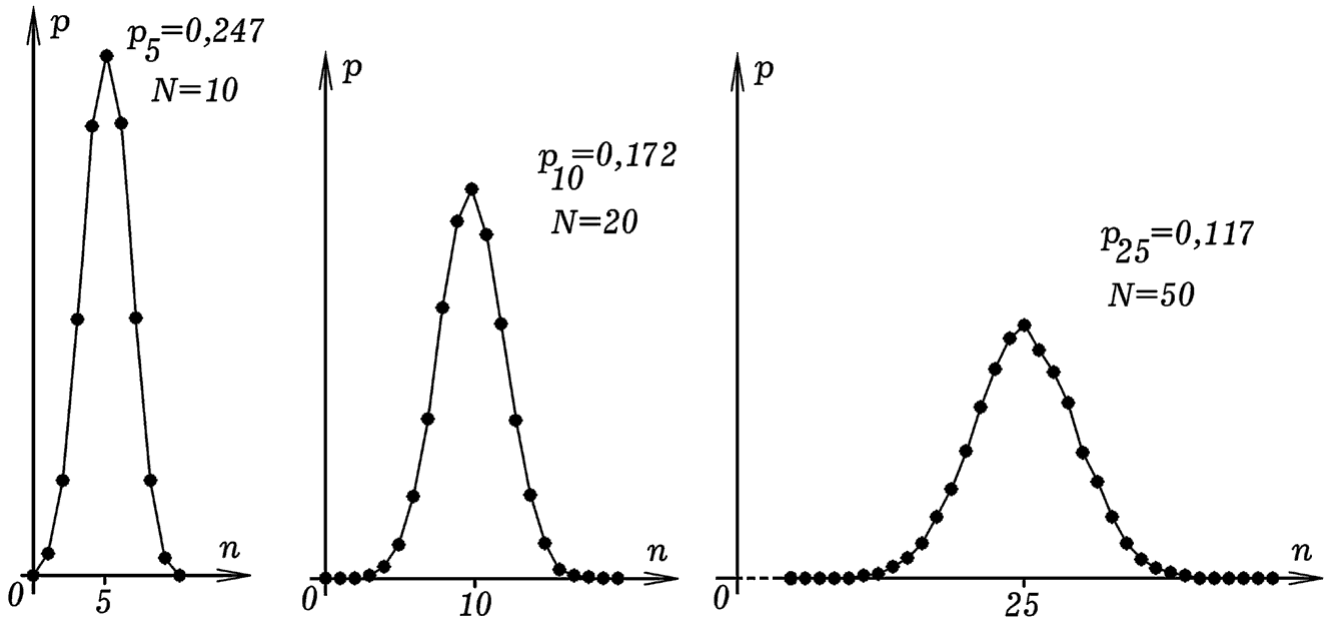

Рис. 10. Зависимость вероятности от числа молекул в левой половине.

Пусть в длинном сосуде находится N молекул газа. Если в левой половине оказалось n частиц, то в правой -- $(N-n)$ частиц. Рассчитаем вероятности различных распределений молекул в левой и правой поло-

винах сосуда и построим график $p(n)$. Для этого будем случайным образом задавать координаты N молекул и подсчитывать их число n в левой половине сосуда. Многократно (более 1000 раз) повторяя эту процедуру, определим вероятности p_1, p_2, p_3, …, p_{N-1}, p_N того, что в левой части окажется 1, 2, 3, …, $N-1$, N молекул. Используется программа ПР-9, результаты для системы из 10, 20 и 50 молекул представлены на рис. 10. Максимальную вероятность имеет **состояние с наибольшей энтропией**, при котором в левой половине находится $N/2$ молекул [2, 7].

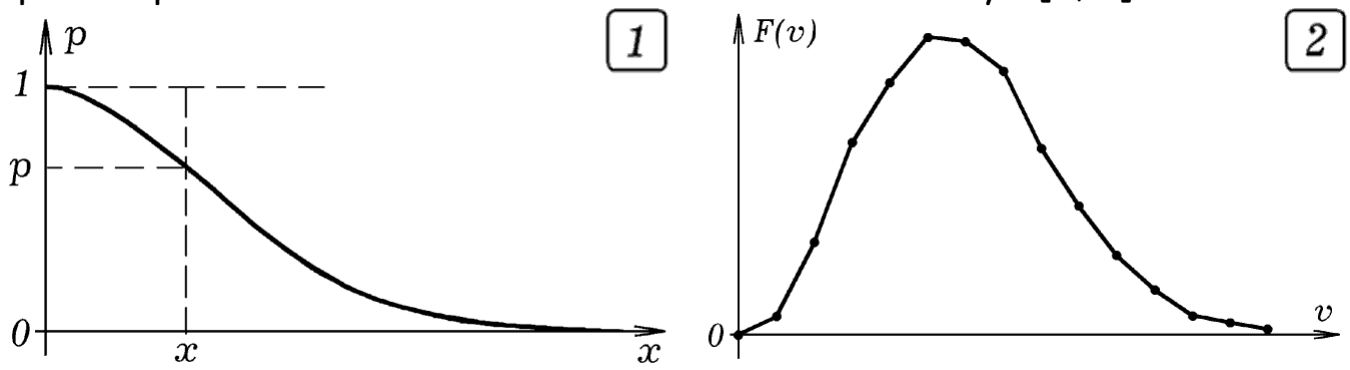

Рис. 11. Распределение молекул по скоростям (закон Максвелла).

Теперь для газа в состоянии термодинамического равновесия получим график распределения молекул по скоростям. Будем исходить из того, все направления равновероятны и проекции скоростей частиц описываются законом нормального распределения Гаусса:

$$F(\upsilon_x) = A\exp(-\upsilon_x^2),\ \ F(\upsilon_y) = A\exp(-\upsilon_y^2),\ \ F(\upsilon_z) = A\exp(-\upsilon_z^2).$$

При этом скорость каждой молекулы $\upsilon = \sqrt{\upsilon_x^2 + \upsilon_y^2 + \upsilon_z^2}$. В цикле задаются значения проекций υ_x, υ_y, υ_z так, чтобы их распределение подчинялось закону Гаусса, после этого вычисляется значение скорости молекулы υ и определяется интервал, к которому принадлежит это значение. Через 10-20 тысяч испытаний строится график распределения молекул по скоростям, который должен соответствовать **закону Максвелла** (рис. 11.2). Для задания случайной величины x, распределенной по закону Гаусса, используется специальная процедура. Генератор случайных чисел вырабатывает случайное число x в интервале от 0 до 5, имеющее равномерное распределение. По закону Гаусса определяется вероятность $p = \exp(-x^2)$ появления данного значения (рис. 11.1). Затем генерируется еще одно случайное число y из интервала $[0;\ 1]$. Если $y < p$, то исходное число x поступает на выход, а иначе все повторяется снова.

9.6. Методы Метрополиса и модельного отжига

В некоторых случаях необходимо определить состояние системы из большого количества частиц, соответствующее глобальному минимуму потенциальной энергии $U(\vec{r}_1,\vec{r}_2,...,\vec{r}_N)=U(x_i,y_i)$, $i=1,2,...,N$. Рассмотрим систему из N частиц, способных перемещаться вдоль оси Oy и соединенных между собой пружинами длиной l_0. Определим координаты y_i частиц, при которых система находится в состоянии устойчивого равновесия. Потенциальная энергия системы равна:

$$U(y_1,y_2,..,y_N)=\sum_{i=2}^{N}\frac{k(l_0-(y_i-y_{i-1}))^2}{2}+\sum_{i=1}^{N}m_i g y_i\,.$$

Будем перебирать частицы системы и изменять их координаты y_i на небольшую случайную величину Δy_i, каждый раз вычисляя новое значение потенциальной энергии U'. Если потенциальная энергия системы в результате смещения i-ой частицы уменьшается, то новое состояние признается энергетически более выгодным и такое изменение конфигурации принимается, а если нет -- отвергается. После этого программа переходит к $i+1$ частице. Одновременно с этим строится график зависимости числа частиц от их высоты. Он получается похожим на экспоненту. Если под частицами понимать слои воздуха равной массы, то эта модель подтверждает, что с ростом высоты концентрация молекул газа уменьшается по экспоненциальному закону. Полученный результат хорошо согласуется с распределением Больцмана $p(E)=A\exp(-E/kT)$, где $p(E)$ -- вероятность нахождения молекулы в состоянии с энергией E, A -- нормировочный множитель.

Иногда **задача поиска глобального минимума** функции $U(x_i,y_i)$ осложняется тем, что эта функция имеет множество локальных минимумов, в которых может "застрять" программа. Рассмотрим совокупность N частиц, которые притягиваются друг к другу и могут образовать двумерную каплю жидкости. Необходимо определить форму капли в невесомости и когда она лежит на горизонтальной поверхности в поле тяжести. Простейший алгоритм состоит в следующем. Сначала программа задает начальную конфигурацию системы, располагая частицы в узлах прямоугольной решетки. Вычисляется потенциальная энергия системы U_1. Затем случайным образом выбирается частица и смещается на небольшую случайную величину, после чего рассчитывается новое значение энергии

U_2. Частицы считаются твердыми шарами, при смещении они не перекрываются. Если потенциальная энергия системы уменьшилась ($U_2 < U_1$), то изменения координат принимаются, а если нет -- отвергаются. Казалось бы такой алгоритм неизбежно должен приводить к правильному решению, но это не так: программа останавливается в одном из локальных минимумов.

Для нахождения глобального минимума используется **алгоритм Метрополиса**, отличающийся следующим. В случае, когда новое значение энергии больше предыдущего ($U_2 > U_1$), эта конфигурация не всегда отвергается, а принимается с вероятностью $p = \exp(-(U_2 - U_1)/kT)$, где T -- "температура" системы [2]. Это позволяет при попадании программы в локальный минимум "потоптаться на месте", а затем "выскочить" из него и попытаться снова найти глобальный минимум функции $U(\vec{r}_1, \vec{r}_2, ..., \vec{r}_N)$.

Развитием алгоритма Метрополиса является **метод модельного отжига** [1]. При этом имитируется отжиг металла, который нагревают до высокой температуры, а затем медленно охлаждают. При высокой температуре частицы колеблются с большой амплитудой, их координаты принимают всевозможные значения, поэтому система может попадать в локальные минимумы. Плавное охлаждение приводит к тому, что амплитуда случайных колебаний атомов (вариаций аргументов) уменьшается, и в результате система оказывается в состоянии соответствующем глобальному минимуму. Сымитировать эту ситуацию можно, используя алгоритм Метрополиса, в котором "температура" T уменьшается пропорционально номеру итерации, либо по экспоненциальному закону. В сложных случаях используется пилообразная зависимость температуры от номера шага, либо адаптивная зависимость когда температура зависит от свойств системы в данный момент времени. Недостаток метода в том, что он требует больших затрат времени.

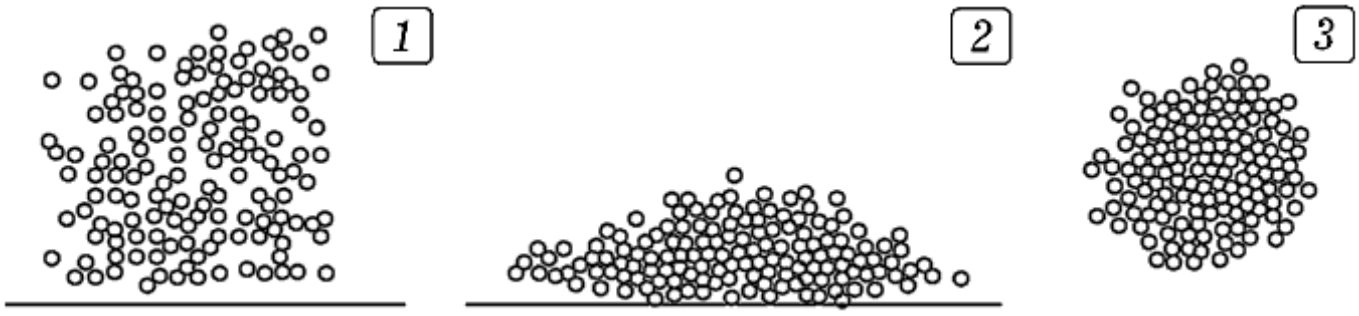

Рис. 12. Поиск состояния с минимальной потенциальной энергией.

Для решения рассмотренной задачи используется программа ПР-10, которая ищет минимум потенциальной энергии системы из $50\text{-}200$ частиц. Используется метод модельного отжига: система медленно "охлаждается", затем осуществляется случайное смещение всех частиц и резкое "нагревание", после этого снова "охлаждение" и т.д. В поле тяжести потенциальная энергия каждой частицы зависит от высоты y_i. Через несколько сотен итераций система переходит в состояние, изображенное на рис. 12.1, затем формируется "капля" на поверхности (рис. 1.2). В невесомости частицы собираются в "шарообразную каплю" (рис. 1.3).

9.7. Модель магнетика Изинга

Еще одной известной моделью термодинамической системы, находящейся в тепловом равновесии с окружающей средой, является модель ферромагнетика, предложенная Изингом в 1924 г. Используется **метод клеточных автоматов**. Представим себе сетку из N горизонтальных и M вертикальных линий. В каждом узле с номерами i и j находится атом, собственный магнитный момент (спин) которого может быть: 1) сонаправлен с осью Oz, и тогда $s_{i,j}=1$; 2) направлен противоположно Oz, что соответствует $s_{i,j}=-1$. Энергия взаимодействия соседних атомов друг с другом равна $-Js_{i,j}s_{i+1,j}$, атомы, расположенные далеко, не взаимодействуют. Кроме того, магнетик может находиться в магнитном поле, вектор напряженности $\vec{h}$ которого параллелен оси Oz. Энергия взаимодействия атома, находящегося в узле (i, j), с внешним магнитным полем равна $hs_{i,j}$. Вся система обладает энергией:

$$E=-J\sum_{i,j}^{N,M}(s_{i,j}s_{i+1,j}+s_{i,j}s_{i-1,j}+s_{i,j}s_{i,j+1}+s_{i,j}s_{i,j-1})+h\sum_{i,j}^{N,M}s_{i,j}.$$

Итак, в двумерной модели магнетика Изинга: 1) кинетическая энергия атомов, находящихся в узлах кристаллической решетки, считается равной 0; 2) предусмотрены только 2 дискретных состояния для спинов атома; 3) учитывается взаимодействие четырех ближайших соседей [4, 7, 10].

Как известно, система стремится перейти в состояние с минимальной потенциальной энергией. При $J>0$ одинаковая ориентация спинов соседних атомов энергетически выгоднее состояний, при которых спины соседних атомов противоположно направлены. Энергия взаимодействия

двух атомов с сонаправленными спинами составляет $-J$, а если спины направлены противоположно, то $+J$. Поэтому состояние с минимальной энергией взаимодействия спинов является ферромагнитным, атомы объединяются в **домены** -- области, в которых магнитный момент (спин) всех атомов ориентирован в одном направлении. Если $J<0$, то система обладает наименьшей энергией в состоянии с антипараллельными спинами, что соответствует антиферромагнетику. Внешнее магнитное поле способствует повороту спина атома так, чтобы он был сонаправлен с напряженностью магнитного поля h.

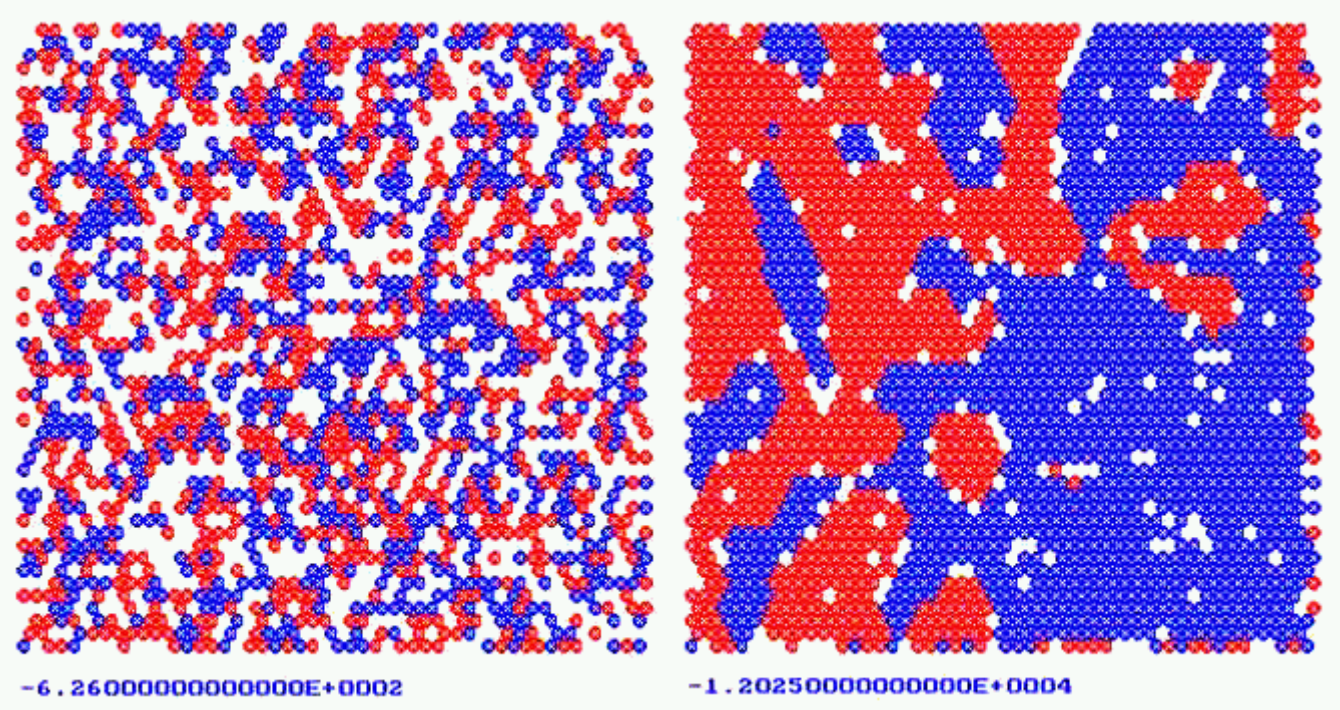

Рис. 13. Возникновение доменов в двумерной модели Изинга.

Процесс моделирования заключается в следующем: 1) задается начальная ориентация всех спинов атомов случайным образом; 2) определяется энергия системы; 3) случайно выбирается один атом, его спин изменяется на противоположный и определяется энергия системы для новой конфигурации; 4) если в результате переориентации спина энергия уменьшилась, то такое изменение конфигурации системы принимается, в противном случае -- отклоняется; 5) возврат к операции 2; если перебраны все атомы, то на экран выводится информация о состоянии системы, затем снова повторяется операция 2. Для поиска состояния с наименьшей энергией можно использовать алгоритм Метрополиса.

Исследуем компьютерную модель ферромагнетика, аналогичную модели Изинга, и отличающуюся от нее: 1) наличием трех состояний у каждого атома ($s_{i,j}=-1,\ 0,\ 1$); 2) учетом влияния восьми соседних ато-

мов. Исходное случайное распределение атомов показано на рис. 13.1. Красные кружки изображают атомы, у которых собственный магнитный момент $s_{i,j}=1$, синие, -- атомы с $s_{i,j}=-1$; в случае, когда $s_{i,j}=0$, атом не изображается. В расчетах учитывается взаимодействие данного атома с восемью ближайшими соседями. Для решения задачи используется программа Пр-11. Результаты моделирования приведены на рис. 13 и 14. Видно, что атомы группируются в **домены** -- области спонтанной намагниченности. Чтобы учесть хаотическое изменение ориентации магнитного момента атомов, обусловленное их тепловым движением, достаточно предусмотреть цикл, в котором перебираются все атомы и с заданной вероятностью инвертируются направление их спинов.

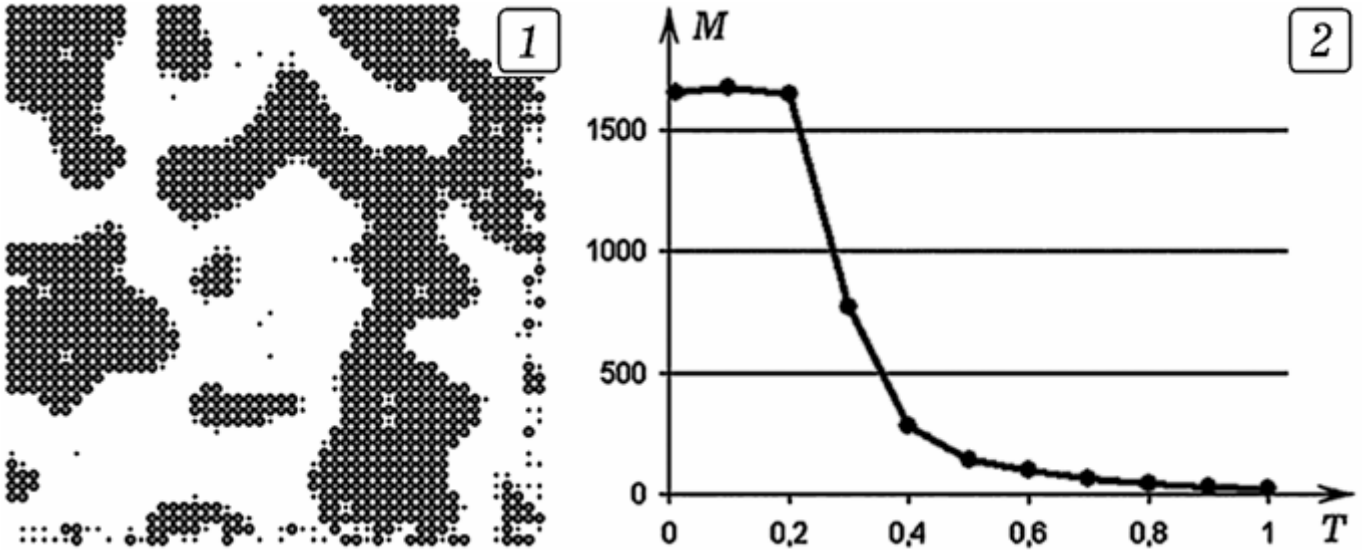

Рис. 14. Зависимость намагничивания от температуры.

При повышении температуры выше критической $T_к$ происходит фазовый переход: ферромагнетик превращается в парамагнетик. Для исследования этого явления необходимо определять среднюю намагниченность M системы при различных температурах T. Ее увеличение должно приводить к росту вероятности p переориентации спинов атомов за 1 шаг по времени. Будем считать, что p связано с T соотношением $p=\exp(-E/kT)$ или $p=\exp(-1/T)$, где T измеряется в условных единицах. В программе ПР-11 для заданного T проводится серия из 100 реализаций и каждый раз определяется средняя намагниченность в пересчете на 1 атом; результаты усредняются. Типичная зависимость M от T представлены на рис. 14.2; резкое падение кривой $M(T)$ соответствует фазовому переходу Изинга.

Приложение

В приложении представлены тексты программ, которые позволяют промоделировать рассмотренные выше явления. Они написаны в средах Borland Pascal 7.0 и Free Pascal 1.0.10.

ПР-1.

```
{$N+} uses crt, graph;
const N=40; dt=0.001; r=3; m=1; Nit=200; Mt=2;
var i,j,k,l,b,DV,MV,Uch : integer;
xsr,t,vv,Temp,p1,p,pp: single;  S: string;
x,v : array[0..N] of single;
BEGIN DV:=Detect; InitGraph(DV,MV,'c:\bp\bgi'); Randomize;
For i:=1 to N do begin x[i]:=13*i;
v[i]:=(1-random(200)/100)*2; end;
Repeat t:=t+dt; Temp:=0;
  For i:=1 to N do begin
   Temp:=Temp+m*sqr(v[i])/2/N; x[i]:=x[i]+v[i]*dt;
   If x[i]>600 then x[i]:=1; If x[i]<0 then x[i]:=599;
{  If x[i]>600 then begin x[i]:=599; v[i]:=-abs(v[i]) end;
   If x[i]<0 then begin x[i]:=1; v[i]:=abs(v[i]) end;} end;
  For i:=1 to N do For j:=1 to N do
   If (x[j]>x[i])and(x[j]-x[i]<2*r) then begin vv:=v[i];
     v[i]:=v[j]; v[j]:=vv; xsr:=(x[i]+x[j])/2; x[i]:=xsr-r;
     x[j]:=xsr+r; end; inc(k);
  For i:=1 to N do For j:=1 to 5 do begin b:=100*j;
  If (x[i]-b)*(x[i-1]-b)<0 then p1:=p1+abs(m*v[i]); end;
  If k mod Nit=0 then begin k:=0; p:=p1/Nit/dt/100;
  {cleardevice; For i:=1 to N do
    circle(10+round(x[i]),200,r);}
  Str(Temp,S); OutTextXY(350,110,S); Str(p,S);
  OutTextXY(350,130,S); p1:=0;
  circle(round(Mt*t),450-round(p),1);
  line(round(Mt*(t-Nit*dt)),450-round(pp),
               round(Mt*t),450-round(p));
  line(round(Mt*(t-Nit*dt))+1,451-round(pp),
               round(Mt*t)+1,451-round(p));
  pp:=p; line(0,450,640,450); end;
until KeyPressed; CloseGraph;
END.
```

ПР-2.

```
{$N+} uses crt, graph;
const N=150; dt=0.001; Mt=0.1; r=5;
Lx=600; Ly=400; m=1; Nit=200;
var i,j,k,l,chislo,DV,MV : integer;
```

```
v1x,v2x,v1y,v2y,xsr,t,vv,Temp,p,p1,pp: single;
x,y,vx,vy : array[0..N] of single; S: string; Label mm, m1;
BEGIN DV:=Detect; InitGraph(DV,MV,'c:\bp\bgi'); Randomize;
For i:=1 to N do begin m1:
  x[i]:=random(Lx); y[i]:=random(Ly);
  For j:=1 to i-1 do
  If sqr(x[i]-x[j])+sqr(y[i]-y[j])<4*r*r+2 then goto m1;
  vx[i]:=1*(8-random(1600)/100);
  vy[i]:=1*(8-random(1600)/100); end;
Repeat  t:=t+dt; Temp:=0;
For i:=1 to N do begin vy[i]:=vy[i]+0*dt;
  x[i]:=x[i]+vx[i]*dt; y[i]:=y[i]+vy[i]*dt;
  Temp:=Temp+m*(sqr(vx[i])+sqr(vy[i]))/2/N;
  If x[i]>Lx-r then begin x[i]:=Lx-r;
    vx[i]:=-abs(vx[i]) end;
  If x[i]<r then begin x[i]:=r; vx[i]:=abs(vx[i]) end;
  If y[i]<r then begin y[i]:=r; vy[i]:=abs(vy[i]) end;
  If y[i]>Ly-r then begin y[i]:=Ly-r;
    vy[i]:=-abs(vy[i]) end; end;
For i:=1 to N do For j:=1 to N do
  If sqr(x[i]-x[j])+sqr(y[i]-y[j])<4*r*r then begin
  v1x:=vx[j]; v1y:=vy[j]; vx[j]:=vx[i]; vy[j]:=vy[i];
  vx[i]:=v1x; vy[i]:=v1y; mm:
  x[i]:=x[i]+vx[i]*dt; y[i]:=y[i]+vy[i]*dt;
  x[j]:=x[j]+vx[j]*dt; y[j]:=y[j]+vy[j]*dt;
 If (i<>j)and(sqr(x[i]-x[j])+sqr(y[i]-y[j])<4*r*r) then
  goto mm; end; inc(k);
For i:=1 to N do
  If (x[i]-Lx/2)*(x[i-1]-Lx/2)<0 then p1:=p1+abs(m*vx[i]);
  If k mod Nit=0 then begin k:=0; p:=p1/Ly/Nit/dt/2;
{ cleardevice; rectangle(20,20,20+Lx,20+Ly);
  For i:=1 to N do circle(20+round(x[i]),20+round(y[i]),r);}
  Str(Temp,S); OutTextXY(350,110,S); Str(p,S);
  OutTextXY(350,130,S); p1:=0;
  circle(round(15*t),460-round(p),1);
  line(round(15*(t-Nit*dt)),460-round(pp),round(15*t),
  460-round(p)); pp:=p; line(0,450,640,450); end;
until KeyPressed; CloseGraph;
END.
```

ТТР-3.

```
{$N+} uses crt, graph;
const  N=50; a=8; dt=0.001;
var Fx,x,v,xx : array[1..N] of single;
f: array[-10..10] of single; Gd,Gm,i,j,k : integer;
t,m,ax,ay,E,l1,l2: single; label Metka, metka1;
Procedure Sila;
begin For i:=1 to N do Fx[i]:=0; For i:=2 to N-1 do
```

```
  begin l1:=abs(x[i]-x[i-1]);   l2:=abs(x[i]-x[i+1]);
  Fx[i]:=8000000*(1/l1/l1/l1/l1-1/l2/l2/l2/l2); end; end;
BEGIN Gd:=Detect; InitGraph(Gd, Gm, 'c:\bp\bgi');
Randomize; m:=1; x[1]:=10; x[N]:=530;
For i:=2 to N-1 do begin  x[i]:=20+10*i;
           v[i]:=random(200)/100-1; end;
Repeat Sila; t:=t+dt;
  For i:=2 to N-1 do begin xx[i]:=x[i]; ax:=Fx[i]/m;
  v[i]:=v[i]+ax*dt; x[i]:=x[i]+v[i]*dt; end; E:=0;
  setcolor(8);
  For i:=1 to N do circle(round(xx[i]/2),100,1);
  setcolor(15);
  For i:=1 to N do circle(round(x[i]),100,1);
  If t>15 then For i:=1 to N do For k:=-8 to 8 do begin
  If (v[i]>k*a)and(v[i]<(k+1)*a) then f[k]:=f[k]+0.01;
  end;
  For k:=-8 to 8 do circle(300+10*k,440-round(f[k]/50),2);
until KeyPressed; CloseGraph;
END.
```

ТТР-4.

```
uses crt, graph; const N=500; dt=0.01;
var D,DV,MV,i,j,a : integer;
x,y,vx,vy,xx,yy: array[1..N] of real;
m: array[1..N] of integer; ver,t: real; entr: real;
BEGIN DV:=Detect; InitGraph(DV,MV,'c:\bp\bgi');
setcolor(8); setbkcolor(15); Randomize;
line(10,400,640,400);
For i:=1 to N do begin x[i]:=random(60)+20;
  y[i]:=random(80)+20; end;
Repeat t:=t+1; a:=80;
For i:=1 to N do begin xx[i]:=x[i]; yy[i]:=y[i];
  x[i]:=x[i]+random(800)/100-4;
  y[i]:=y[i]+random(800)/100-4;
  If t>500 then a:=120; If t>1500 then a:=200;
  If (x[i]<20)or(x[i]>a) then x[i]:=xx[i];
  If (y[i]<20)or(y[i]>100) then y[i]:=yy[i]; end;
setcolor(15);
For i:=1 to N do circle(round(xx[i]),round(yy[i]),1);
setcolor(8);
For i:=1 to N do circle(round(x[i]),round(y[i]),1);
entr:=0; For j:=1 to 20 do begin m[j]:=0;
For i:=1 to N do
  If (x[i]>20*j)and(x[i]<20+20*j) then m[j]:=m[j]+1;
entr:=entr-(m[j]+0.01)/(N+0.01)*ln((m[j]+0.01)/(N+0.01));
end; circle(round(10+t/10),400-round(entr*50),1);
until Keypressed; CloseGraph;
END.
```

ПР-5.

```
{$N+} uses crt, graph;
const N=60; m=1; dt=0.00001; Lx=400; Ly=400; Nit=100;
var Fx,Fy,x,y,vx,vy,xx,yy: array[0..N+1] of single;
    vir,pvir,Tk,Tem,a,t,ax,ay,F,l,s: single;
    Gd,Gm,i,j,k: integer; C:string;
Procedure Sila;
begin For i:=1 to N do begin Fx[i]:=0; Fy[i]:=0; end;
For i:=1 to N do for j:=1 to N do begin
  If j<>i then begin l:=sqrt(sqr(x[i]-x[j])+sqr(y[i]-y[j]));
  If l<8 then l:=1; F:=(1000000/l/l/l-500000/l/l);
  Fx[i]:=Fx[i]+F*(x[i]-x[j])/l;
  Fy[i]:=Fy[i]+F*(y[i]-y[j])/l; end; end; end;
BEGIN Gd:=Detect; InitGraph(Gd,Gm,'c:\bp\bgi');
setbkcolor(15); setcolor(8); Randomize; a:=1;
For i:=1 to N do begin x[i]:=random(200)+50;
  y[i]:=random(200)+50; vy[i]:=600*(random(100)/10-5);
  vx[i]:=600*(random(100)/10-5); end;
Repeat
  t:=t+dt; If (t>0.1)and(t<0.5) then Tk:=1E+4 else Tk:=1E+6;
  If Tem>Tk then begin a:=0.9999; circle(6,6,2); end
    else a:=1.0001;
  For i:=1 to N do begin ax:=Fx[i]/m; ay:=Fy[i]/m;
    x[i]:=x[i]+vx[i]*dt+ax*dt*dt/2;
    y[i]:=y[i]+vy[i]*dt+ay*dt*dt/2;
    vx[i]:=a*vx[i]+ax*dt/2; vy[i]:=a*vy[i]+ay*dt/2;
    vir:=vir+x[i]*Fx[i]+y[i]*Fy[i];
  If x[i]<50 then x[i]:=248; If x[i]>250 then x[i]:=52;
  If y[i]<50 then y[i]:=248; If y[i]>250 then y[i]:=52; end;
  Sila; Tem:=0;
For i:=1 to N do begin ax:=Fx[i]/m; ay:=Fy[i]/m;
  vx[i]:=a*vx[i]+ax*dt/2; vy[i]:=a*vy[i]+ay*dt/2;
  Tem:=Tem+m*(sqr(vx[i])+sqr(vy[i]))/2/N; end; inc(k);
If k mod Nit=0 then begin cleardevice; k:=0;
  rectangle(50,50,250,250);
  For i:=1 to N do circle(round(x[i]),round(y[i]),5);
  pvir:=N*Tem/Lx/Ly+0.5*vir/Lx/Ly/Nit; vir:=0;
  Str(Tem,C); OutTextXY(310,410,C);
  Str(pvir,C); OutTextXY(310,430,C); end;
until KeyPressed; CloseGraph;
END.
```

ПР-6.

```
{$N+}uses crt,graph; const N=140; dt=0.0003;
var Fx,Fy,x,y,vx,vy: array[1..N] of single;
  t,Gd,Gm,i,j: integer; a,m,k,F,l,E : single; S: string;
Procedure Sila;
```

```
begin For i:=1 to N do begin Fx[i]:=0; Fy[i]:=0; end;
For i:=1 to N do For j:=1 to N do If j<>i then begin
  l:=sqrt(sqr(x[i]-x[j])+sqr(y[i]-y[j]));
  If (l>0)and(l<10) then F:=40000/l else F:=0;
  Fx[i]:=Fx[i]+F*(x[i]-x[j])/l;
  Fy[i]:=Fy[i]+F*(y[i]-y[j])/l; end; end;
BEGIN Gd:=Detect; InitGraph(Gd,Gm,'c:\bp\bgi');
setbkcolor(white); Randomize; a:=1;
For i:=1 to N do begin y[i]:=104+random(194);
If i<N/2 then x[i]:=102+random(198)
         else x[i]:=302+random(198);
vx[i]:=random(800)/10-40; vy[i]:=random(800)/10-40; end;
Repeat Sila; inc(t); E:=0; m:=0.01;
For i:=1 to N do begin If i=N then m:=0.2 else m:=0.01;
  vx[i]:=a*vx[i]+Fx[i]/m*dt; vy[i]:=a*vy[i]+Fy[i]/m*dt;
  x[i]:=x[i]+vx[i]*dt; y[i]:=y[i]+vy[i]*dt;
  E:=E+m*(sqr(vx[i])+sqr(vy[i]))/2;
  If x[i]>500 then begin x[i]:=499; vx[i]:=-vx[i]; end;
  If x[i]<100 then begin x[i]:=101; vx[i]:=-vx[i]; end;
  If y[i]>300 then begin y[i]:=299; vy[i]:=-vy[i]; end;
  If y[i]<100 then begin y[i]:=101; vy[i]:=-vy[i]; end; end;
  If E>5E+5 then a:=0.999 else a:=1.001;
  If t mod 5=0 then begin cleardevice; t:=0;
  setcolor(8); rectangle(96,96,504,304);
  For i:=1 to N do If i<N/2 then begin
    setcolor(9); circle(round(x[i]),round(y[i]),3); end
      else begin setcolor(2); If i=N then setcolor(5);
        circle(round(x[i]),round(y[i]),5); end;
  circle(round(x[N]),round(y[N]),2);
  str(E,S); OuttextXY(250,350,S); end;
until KeyPressed; CloseGraph;
END.
```

ПР-7.

```
{$N+}uses crt, graph;
const N=50; M=100;
var i,j,k,l,Gd,Gm,i1,j1,t,M1,MM: integer; sl,p,Kon: single;
s: array [-1..N+1,-1..M+1] of integer;
Procedure RRR;
begin sl:=random(100)/100;
If (s[i-1,j]=0)and(sl<0.25) then
         begin s[i,j]:=0; s[i-1,j]:=1; end;
If (s[i+1,j]=0)and(sl>0.24)and(sl<0.5) then
         begin s[i,j]:=0; s[i+1,j]:=1; end;
If (s[i,j-1]=0)and(sl>0.5)and(sl<p) then
         begin s[i,j]:=0; s[i,j-1]:=1; end;
If (s[i,j+1]=0)and(sl>p) then
```

```
    begin s[i,j]:=0; s[i,j+1]:=1; end; end;
BEGIN M1:=30; Randomize;
Gd:=Detect; InitGraph(Gd,Gm,'c:\bp\bgi');
For i:=1 to N do For j:=1 to 10 do begin
s[i,j]:=1; s[i,M]:=1; end;
{For k:=1 to 2000 do begin i:=round(random(N-5))+2;
j:=round(random(M-5))+2; s[i,j]:=1; end;}
For j:=1 to M do begin s[1,j]:=1; s[N,j]:=1; end;
For i:=1 to N do begin s[i,1]:=1; s[i,M]:=1; end;
Repeat inc(t);
If t<100 then begin MM:=M1;
  For i:=1 to N do s[i,M1]:=1; end else MM:=M;
If t=100 then For i:=2 to N-1 do s[i,M1]:=0;
For k:=1 to 10000 do begin
  i:=round(random(N-2))+2; j:=round(random(MM-2))+2;
  If s[i,j]=1 then begin Kon:=0;
    For i1:=i-2 to i+2 do For j1:=j-2 to j+2 do
    Kon:=Kon+s[i1,j1];
    If Kon>2 then p:=0.85 else p:=0.65; RRR; end;
end;
inc(l); If l mod 20=0 then l:=0;
delay(200); cleardevice;
For i:=1 to N do For j:=1 to M do
If s[i,j]=1 then begin circle(10+4*i,450-4*j,2);
circle(10+4*i,450-4*j,1); end;
until KeyPressed; CloseGraph;
END.
```

ПР-8.

```
uses crt, graph; const Kol_shag=150; Kol_isp=5000;
var n: array[-20..20]of integer; x:array[1..55]of real;
a,b,i,j,k,DV,MV,EC:integer; D,L,Y,z,S: real; St:string;
BEGIN DV:=Detect; InitGraph(DV,MV,'c:\bp\bgi');
For i:=1 to 55 do x[i]:=random(10000)/10000;
a:=55; b:=24; j:=0; S:=0;
Repeat z:=0; inc(j);
  For k:=1 to Kol_shag do begin
  If x[56-a]>=x[56-b] then Y:=x[56-a]-x[56-b] else
    Y:=x[56-a]-x[56-b]+1;
  For i:=1 to 54 do x[i]:=x[i+1]; x[55]:=y;
  If x[55]<0.5 then z:=z-(0.5-y) else z:=z+(y-0.5); end;
  For i:=-15 to 15 do If (z>=i)and(z<i+1) then inc(n[i]);
  S:=z*z+S;
until (j>Kol_isp)or(KeyPressed); D:=S/(Kol_isp-1);
str(D,St); OuttextXY(50,50,St);
For i:=-15 to 15 do circle(330+i*10,450-round(n[i]/3),2);
Repeat until (KeyPressed); CloseGraph;
END.
```

ТР-9.

```
Uses crt, graph; const N_mol=50; N_isp=15000;
Var  n: array[0..N_mol+1] of integer;
l,p,xx : real; s,ii,i,j,k,z,t,f,DV,MV : integer;
BEGIN Randomize; clrscr;
Repeat k:=k+1; s:=0;
  For i:=1 to N_mol do begin xx:=random(1000)/1000;
  If xx<0.5 then s:=s+1; end; {writeln(s);}
  For i:=0 to N_mol do if s=i then inc(n[i]);
until k>N_isp;
For i:=0 to N_mol do writeln(i,' ',n[i]/N_isp,'|');
DV:=Detect; InitGraph(DV,MV,'c:\bp\bgi');
line(0,400,10*N_mol,400);
For i:=0 to N_mol do circle(10+10*i,400-round(n[i]/10),2);
Repeat until KeyPressed; CloseGraph;
END.
```

ТР-10.

```
{$N+} uses crt, graph;
const  N=144;  m=1; dt=0.02; Lx=200; Ly=200; Km=200;
var x,y,xx,yy: array[-2..N+2] of real;
aa,bb,min,sl,U,U1,a,t,l,p,s: single;
g,q,Gd,Gm,i,j,k: integer; C: string; Label m1,m2,m3;
Procedure Energy;
begin U:=0; For i:=1 to N do For j:=1 to N do begin
 If j<>i then begin s:=abs(300-y[i]);
 l:=sqrt(sqr(x[i]-x[j])+sqr(y[i]-y[j]));
 U:=U-1500/l/l-0.1*y[i]+500/s/s; end; end; end;
BEGIN Gd:=Detect; InitGraph(Gd,Gm,'c:\bp\bgi'); Randomize;
For i:=1 to N do begin k:=round(i/12);
  x[i]:=250+15*i-k*180; y[i]:=100+15*k; end;
Repeat q:=round(random(n*10)/10);
For g:=1 to 5 do begin m1: aa:=x[q]; bb:=y[q];
  x[q]:=aa+(random(200)/100-1)*15;
  y[q]:=bb+(random(200)/100-1)*15;
  If Keypressed then goto m2;
  For j:=1 to N do If ((q<>j)and(sqr(x[q]-x[j])+
    sqr(y[q]-y[j])<90))or(y[q]>299) then
    begin x[q]:=aa; y[q]:=bb; goto m1; end; Energy;
 If U>min then begin p:=1/(1+exp(t*(U-min)*6/Km));
   circle(10,10,3); sl:=random(100);
   If sl<p then min:=U else begin x[q]:=aa; y[q]:=bb; end;
end;
If U<=min then min:=U;
If U<=min then begin cleardevice; line(0,300,600,300);
  For i:=1 to N do circle(round(x[i]),round(y[i]),5);
  Str(p,C); OutTextXY(310,350,C); Str(k,C);
  OutTextXY(310,10,C); Str(min,C); OutTextXY(310,410,C);
```

```
  Str(U,C); OutTextXY(310,430,C); end; end; inc(k);
If k mod Km=0 then For g:=1 to 6 do For q:=1 to N do
  begin m3: aa:=x[q]; bb:=y[q];
    x[q]:=aa+(random(200)/100-1)*2;
    y[q]:=bb+(random(200)/100-1)*2;
    If Keypressed then goto m2;
    For j:=1 to N do If ((q<>j)and(sqr(x[q]-x[j])+
     sqr(y[q]-y[j])<90))or(y[q]>299) then begin x[q]:=aa;
       y[q]:=bb; goto m3; end;
  k:=0; min:=0.6*min; t:=0; end;
until KeyPressed; m2: CloseGraph;
END.
```

ТТР-11.

```
{$N+}uses crt, graph;
const N=50; M=50; jj=1; Temp=0.2; N_shag=100;
var i,j,k,l1,u,Gd,Gm: integer;
Sm,sl,p,h,E,EE,dE,x,a,fi,fi1,fi2 : single;
s : array [1..N,1..M] of single; C: string; Label metka;
Procedure Energy;
Var i,j : integer;
begin E:=0; dE:=0;
For i:=2 to N-1 do For j:=2 to M-1 do begin
  E:=E-jj*(s[i,j]*s[i+1,j]+s[i,j]*s[i-1,j]+s[i,j]*s[i,j+1]+
  s[i,j]*s[i,j-1]+0.7*(s[i,j]*s[i+1,j+1]+s[i,j]*s[i-1,j-1]+
  s[i,j]*s[i-1,j+1]+s[i,j]*s[i+1,j-1])); end;
For i:=2 to N-1 do For j:=2 to M-1 do begin h:=2;
  {If (i>0.6*N)and(j>0.6*M) then h:=-2.5;
  If (i<0.4*N)and(j<0.4*M) then h:=2.5;}
  dE:=dE-h*s[i,j]; end; E:=E+dE; end;
Procedure Draw;
begin cleardevice; For i:=1 to N do For j:=1 to M do
  begin If s[i,j]=1 then setcolor(red);
  If s[i,j]=-1 then setcolor(blue);
  If s[i,j]=0 then setcolor(green); circle(7*i,7*j,1);
  circle(7*i,7*j,2); circle(7*i,7*j,3); end;
  Str(Sm,C); OutTextXY(310,410,C); Sm:=0; end;
BEGIN Gd:=Detect; InitGraph(Gd,Gm,'c:\bp\bgi'); Randomize;
For i:=1 to N do For j:=1 to M do begin s[i,j]:=0;
  p:=Random(100)/100; If p<0.33 then s[i,j]:=-1;
  If p>0.66 then s[i,j]:=1; end; Draw;
Repeat inc(l1);
For k:=1 to round(N*M/10) do begin Energy; EE:=E;
  i:=1+round(random(N-1)); j:=1+round(random(M-1));
  If s[i,j]=1 then begin u:=1; s[i,j]:=0; goto metka; end;
  If s[i,j]=-1 then begin u:=-1; s[i,j]:=0; goto metka; end;
  If (s[i,j]=0)and(random(100)>50) then s[i,j]:=-1
                                   else s[i,j]:=1;
```

```
  u:=0; metka: Energy; If (E>EE) then s[i,j]:=u; end;
For i:=1 to N do For j:=1 to M do begin sl:=random(100)/100;
  If sl<exp(-1/Temp) then s[i,j]:=-s[i,j];
  Sm:=Sm+s[i,j]/N_shag; end;
If l1 mod 10{N_shag}=0 then Draw;
until KeyPressed; CloseGraph;
END.
```

ЛИТЕРАТУРА

1. Булавин Л.А., Выгорницкий Н.В., Лебовка Н.И. Компьютерное моделирование физических систем. –- Долгопрудный: Издательский Дом "Интеллект", 2011. –- 352 с.
2. Гулд Х., Тобочник Я. Компьютерное моделирование в физике: В 2-х частях. Часть 1. –- М.: Мир, 1990. –- 350 с.
3. Данилов О.Е. Компьютерное моделирование движения молекул газа // Проблемы учебного физического эксперимента: Сборник научных и методических работ. Выпуск 2. –- Глазов: ГГПИ, 1996. –- С. 78 – 80.
4. Кунин С. Вычислительная физика. –- М.: Мир, 1992. –- 518 с.
5. Майер Р.В. Задачи, алгоритмы, программы. [Электронный ресурс] /URL: http://maier-rv.glazov.net, http://komp-model.narod.ru.
6. Майер Р.В. Компьютерное моделирование физических явлений. –- Глазов, ГГПИ: 2009. –- 112 с.
7. Поршнев С.В. Компьютерное моделирование физических процессов в пакете MATLAB. –- М.: Горячая линия - Телеком, 2003. -- 592 с.
8. Рудяк В.Я. Математические модели природных явлений и технологических процессов. В 2 ч. – Новосибирск: Изд-во НГТУ, 2003. – Ч. 1. – 181 с.
9. Федоренко Р.П. Введение в вычислительную физику: Учеб. пособие для вузов. –- М.: Изд-во Моск. физ.-техн. ин-та, 1994. –- 528 с.
10. Эксперимент на дисплее. Первые шаги вычислительной физики. – М.: Наука, 1989. –- 175 с.
11. Giordano N.J. Computational Physics. -- New Jersey, Prentice Hall, 1997, 419 p.
12. Woolfson M.M., Pert G.J. An Introduction to Computer Simulation. -- Oxford University Press, 1999, 311 p.