

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет радиофизики и электроники

КАФЕДРА СИСТЕМНОГО АНАЛИЗА

*В.М. Лутковский, П.В. Назаров*

**Учебно-методический комплекс по специальному курсу**

**«МОДЕЛИРОВАНИЕ ПРОЦЕССОВ И СИСТЕМ»**

для студентов 4-го курса специальностей

«Радиофизика» и «Физическая электроника»

Минск 2010

## Содержание

<b>СПИСОК СОКРАЩЕНИЙ</b> .....	<b>2</b>
<b>ПРЕДИСЛОВИЕ</b> .....	<b>3</b>
<b>ВВЕДЕНИЕ: ЦЕЛИ, ЗАДАЧИ, МЕТОДЫ</b> .....	<b>4</b>
<b>I. МОДЕЛИРОВАНИЕ ДИНАМИЧЕСКИХ СИСТЕМ</b> .....	<b>6</b>
1. Динамическая система.....	6
2. Метод молекулярной динамики.....	9
<b>II. ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ АНАЛИЗ И МОДЕЛИРОВАНИЕ СЛОЖНЫХ СИСТЕМ</b> .....	<b>12</b>
3. Сложные системы.....	12
4. Объектная модель.....	19
<b>III. ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА МОДЕЛИРОВАНИЯ</b> .....	<b>21</b>
5. Универсальные программные средства .....	21
6. Специализированные инструментальные средства .....	26
<b>IV. МОДЕЛИРОВАНИЕ ФОТОФИЗИЧЕСКИХ ПРОЦЕССОВ</b> .....	<b>28</b>
7. Ветвящийся процесс (процесс рождения и гибели) .....	28
8. Имитационное моделирование дискретных процессов.....	30
<b>V. ПРИМЕРЫ МОДЕЛЕЙ СИСТЕМ</b> .....	<b>41</b>
9. Модели электронных приборов .....	41
10. Системы сбора данных.....	47
11. Системы массового обслуживания .....	50
12. Система охранной сигнализации .....	52
Заключение .....	54
Литература .....	55
Приложение 1 Лабораторная работа № 1 .....	74
Приложение 2.Лабораторная работа № 2 .....	57
Приложение 3. Лабораторная работа № 3 .....	58
Приложение 4. Лабораторная работа № 4 .....	58
Приложение 5. Лабораторная работа № 5 .....	59
Приложение 6. Код программы MD2 (Pascal) – к лаб. работе № 2.....	62
Пример реализации метода молекулярной динамики на основе алгоритма Верле	
Список сокращений	

ВП – ветвящийся процесс

ЛФД – лавинный фотодиод;

МКП – микроканальная пластина;

СМО – система массового обслуживания;

ТМО – теория массового обслуживания,

ФЭУ – фотоэлектронный умножитель;

МРІ (The Message Passing Interface) - интерфейс передачи сообщений.

## ПРЕДИСЛОВИЕ

Моделирование – необходимый этап цикла проектирования систем. Современные методы моделирования позволяют значительно ускорить процесс создания современных электронных приборов и систем. Развитие компьютерных технологий приводит нас к новому взгляду на исследование процессов в сложных системах, к числу которых относятся и современные электронные и микроэлектронные приборы. Важным инструментом анализа сложных явлений в физике и технике остается численное моделирование (вычислительный эксперимент).

## ВВЕДЕНИЕ: ЦЕЛИ, ЗАДАЧИ, МЕТОДЫ

Приступая к изучению спецкурса, необходимо ответить на вопросы:

**Что, зачем и как изучается в этом курсе?**

**ii) Что** – (см. стр. 2 - содержание)

**i) Зачем** – поясняется следующими примерами проблемных ситуаций.

### Примеры применения моделирования процессов и систем

Проблема	Преимущества моделирования
Не удается найти подходящего технического решения	Моделирование гипотетической системы
Решение существует, но оно неэффективно	Оптимизация через моделирование
Реальную систему (или ее часть) очень трудно, долго или опасно изучать в различных условиях	Упрощение или ускорение испытаний всей системы или ее подсистем (ледоколы, реакторы, ускорители, энергетические системы, микроорганизмы и др.)
Высокие затраты на разработку системы	Снижение затрат на разработку (модель – упрощение сложной системы)
Подготовка экипажей для военной техники, орбитальных и межпланетных экспедиций	Снижение риска, материальных и временных затрат при использовании тренажеров

**Цели и задачи** моделирования вытекают из основных определений. Напомним наиболее важные определения.

**Модель** – (от лат. *modulus* – мера, образец) – любой образ, аналог (мысленный или условный: изображение, описание, схема, чертеж, график, карта и т.п.) того или иного объекта, процесса или явления ("оригинала"). (Энциклопедический словарь). Другими словами, моделью может быть любой другой объект, отдельные свойства которого полностью или частично совпадают со свойствами исходного. Модель создается ради исследований, которые на реальном объекте проводить либо невозможно, либо дорого, либо просто неудобно.

**Моделирование** – исследование тех или иных явлений, процессов или систем объектов путем построения и изучения их моделей; использование моделей для определения или уточнения характеристик и улучшения способов построения вновь конструируемых объектов.

**Методы моделирования** могут быть различными: физическое, математическое, аналоговое, цифровое, компьютерное.

Физическая модель - обычно уменьшенная копия реального объекта (например, модель или макет самолета). Аналоговое моделирование основано на замене исходного объекта объектом другой физической природы, обладающим аналогичным поведением. Например, электрические цепи позволяют имитировать поведение механических систем.

**Компьютерное моделирование** можно рассматривать как продолжение и развитие математического моделирования.

### ***Какие требования предъявляются к модели процесса или системы?***

Модель должна с заданной точностью отражать поведение реального объекта в реальных условиях. "Модель должна быть простой, но не проще (разумного предела)". Очевидно, такая модель должна помогать более эффективному решению наших проблем.

Если же решение проблемы легко можно найти на основании опыта или перенести из другой области, то можно обойтись без компьютерного моделирования. ***Методы изучения спецкурса.*** Приведем аналогию, проясняющую особенность методики изучения данного спецкурса. Прежде всего, зададим себе вопрос: ***какую модель можно использовать для описания процесса обучения?***

Из множества различных вариантов мы рассмотрим два: «*наполнить сосуд*» и «*зажечь светильник*». За три года учебы студенты получают огромную дозу информации, которые используются не всегда эффективно. Наша задача состоит в том, чтобы сделать эти знания активными, образно говоря, зажечь светильник этих знаний. Смысл рассмотренного примера в том, что процесс обучения может быть направлен на решение различных задач. Мы будем акцентировать внимание на максимальном применении того, что изучено прежде, а за новой информацией обращаться по мере необходимости.

Таким образом, методика изучения данного спецкурса направлена на приобретение практического опыта разработки и применения моделей процессов в радиофизических системах. Поэтому лекционный материал содержит лишь необходимый минимум информации, необходимой для выполнения лабораторных работ. При этом своевременное выполнение и защита лабораторных работ – ***основной критерий успешности.***

***Цель лабораторных работ*** – получение практического опыта моделирования процессов в радиофизических системах, электронных и микроэлектронных приборах. Методика проведения лабораторных работ предусматривает обязательную *самостоятельную подготовку*.

Первый этап самостоятельной подготовки - повторение материала по физическим основам электроники, физике полупроводниковых приборов, а также более углубленное изучение специальных методов моделирования электронных приборов и их характеристик. С этой целью студентам предлагаются контрольные вопросы и список рекомендуемой литературы.

До начала занятий в компьютерном классе студенты должны согласовать требования к выполнению задания (тип прибора, моделируемые характеристики, степень детализации физических процессов). В большинстве последующих заданий используются результаты предыдущих, поэтому вопросы выбора структуры моделей и системы условных обозначений имеют большое значение.

*Наряду с лабораторными работами имеется возможность разработки оригинальных проектов. Методика организации выполнения проектов рассмотрена в работе "Метод проектов"*

***iii) Таким образом, раскрывается ответ на вопрос: Как изучается данный спецкурс ?***

# I. МОДЕЛИРОВАНИЕ ДИНАМИЧЕСКИХ СИСТЕМ

## 1. Динамическая система

Изучению динамической системы как математической модели уже более ста лет. Возникновение этого направления прежде всего было связано с небесной механикой.

В данной лекции прежде всего рассматриваются модели *линейных динамических систем* (механических, электрических и др.), которые могут быть описаны системой дифференциальных уравнений. Абстрагируясь от конкретной физической природы объекта, о нем можно говорить как о динамической системе, если заданы:

- динамические переменные
- состояние системы, определяемое динамическими переменными
- оператор эволюции системы

Динамическая система – абстракция, как и «материальная точка», «идеальный газ» и другие.

Динамические системы могут быть описаны системами обыкновенных дифференциальных уравнений, уравнениями в частных производных, либо разностными уравнениями на детерминистской или стохастической основе. Динамические свойства систем в этих уравнениях определяются производными по времени или соответствующими им разностными выражениями. Системы, описываемые дифференциальными уравнениями в частных производных, могут быть аппроксимированы обыкновенными дифференциальными уравнениями, которые содержат только производные по времени.

**Пространство состояний.** Известно, что любое из обыкновенных дифференциальных уравнений порядка  $r$  можно преобразовать в систему дифференциальных уравнений первого порядка. Система из  $n$  дифференциальных уравнений первого порядка определена полностью лишь в том случае, когда заданы все коэффициенты и известны  $n$  начальных условий.

Рассмотрим линейную систему, описываемую следующим векторным уравнением состояния:

$$\frac{dx}{dt} = Ax + Bu \quad (*)$$

где  $x = n \times 1$ -вектор состояния;  $u = m \times 1$  - вектор возмущающих воздействий, или входной вектор, компоненты которого могут быть независимыми функциями времени;  $A, B$  – матрицы коэффициентов.

Рассмотренное векторное дифференциальное уравнение является уравнением состояния динамической системы.

Начальные условия образуют  $n$ -мерный вектор, который полностью (и точно) определяет состояние системы, описываемой названными уравнениями, в начальный момент времени  $t_0$  (предполагается, что все входные или возмущающие воздействия известны с момента  $t_0$  и далее). Указанный вектор называется вектором состояния системы в момент времени  $t_0$ , а его компоненты называются переменными состояния. Вектор состояния может быть образован различными комбинациями  $n$  переменных состояния.

Для определения зависимости состояния системы от времени необходимо проинтегрировать соответствующую систему дифференциальных уравнений с учетом заданного вектора возмущающих воздействий.

В качестве примера рассмотрим модель «хищник-жертва», описываемую системой дифференциальных уравнений Лотка-Вольтера.

Модель взаимодействия "хищник-жертва" независимо предложили в 1925-1927 гг. Лотка и Вольterra. Два (или три) дифференциальных уравнения

$$\begin{aligned} dy_1/dt &= [3*y(1) - 0.3*y(1)*y(2)]; \\ dy_2/dt &= [-1.2*y(2) + 1.2/25*y(1)*y(2)]; \end{aligned}$$

$$\begin{aligned} dy_1/dt &= [3*y(1)*(1 - 1/10*y(2) + 1/15*y(3))]; \\ dy_2/dt &= [1.2*y(2)*(1 + 1/20*y(1) - 1/25*y(3))]; \\ dy_3/dt &= [2*y(3)*(1 - 1/30*y(1) + 1/35*y(2))]; \end{aligned}$$

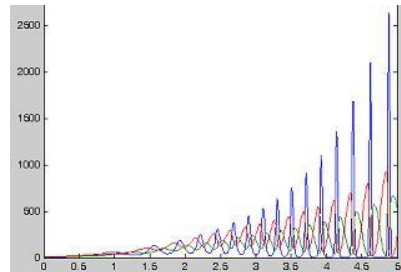


Рис. \*\*

моделируют временную динамику численности двух биологических популяций жертвы  $y_0$  и хищника  $y_1$ . Предполагается, что жертвы размножаются с постоянной скоростью  $C$ , а их численность убывает вследствие поедания хищниками. Хищники же размножаются со скоростью, пропорциональной количеству пищи (с коэффициентом  $r$ ) и умирают естественным образом (смертность определяется константой  $D$ ). Модель замечательна тем, что в такой системе наблюдаются циклическое увеличение и уменьшение численности и хищника (рис. \*\*), и жертвы, так часто наблюдаемое в природе.

Фазовый портрет системы представляет собой концентрические замкнутые кривые, окружающие одну стационарную точку, называемую центром. Как видно, модельные колебания численности обеих популяций существенно зависят от начальных условий - после каждого периода колебаний система возвращается в ту же точку. Динамические системы с таким поведением называют *негрубыми*

**Передаточная функция.** Для решения системы уравнений (\*) применяют метод преобразований Лапласа [Понтрягин]. Система дифференциальных уравнений

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \quad (1)$$

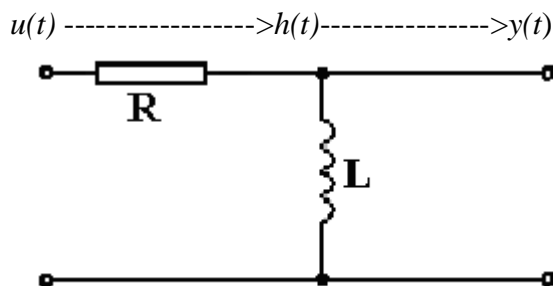
$x(t)$  - вектор состояния;  $u(t)$  - вектор возмущающих воздействий;  $A$  - матрица определяемая системой дифференциальных уравнений;  $B$  - матрица определяющая управляемость системы;  $C$  - матрица определяющая наблюдаемость системы.

С помощью преобразования Лапласа [1,2], можно перейти от системы дифференциальных уравнений к системе алгебраических уравнений:

$$Y(s) = H(s)u(s)$$

где  $s$  - комплексная переменная.

**Пример** линейной системы:



Передаточная функция такой системы имеет вид:

$$H(s) = \frac{sL}{sL + R}.$$

В системе MATLAB модели динамических систем можно представить в следующих формах:

- State-space - пространство состояний
- Transfer function - передаточная функция
- Zero-pole - нулей-полюсов (нули – корни алгебраического выражения в числителе, полюса – корни выражения в знаменателе передаточной функции).

**Нелинейная динамика.** В качестве примера рассмотрим систему трех нелинейных уравнений первого порядка (Кузнецов, Lorenz, 1963):

$$dx/dt = \sigma(y-x), \quad dy/dt = r*x - y - x*z, \quad dz/dt = -b*z + x*y, \quad (***)$$

Такая система описывает динамику нескольких физических систем – конвекцию в слое, конвекцию в кольцевой трубке, одномодовый лазер. Если взять значения параметров  $\sigma = 10$ ;  $b=8/3$ ,  $r=28$  и провести численное решение уравнений (\*\*\*), то обнаруживается, что в системе устанавливается хаотический автоколебательный режим. На рис. Приводятся зависимости динамических переменных  $x$ ,  $y$ ,  $z$  от времени

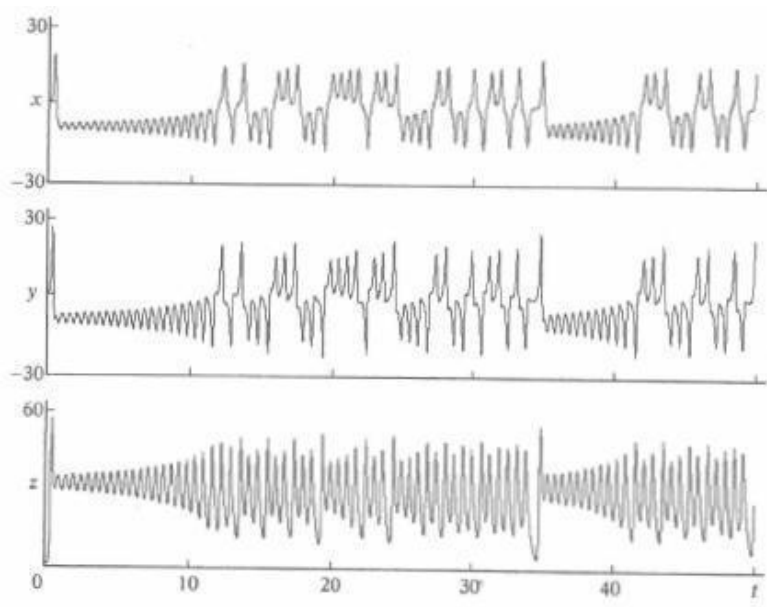


Рис. 4.1 Динамика системы Лоренца

1. Кузнецов С.П. Динамический хаос (курс лекций). – М.: Изд-во Физико-математической литературы, 2001. –296 с.

### Литература

2. Гроп Д. Идентификация систем. М.: Мир.
3. Понтрягин Л.С. Дифференциальные уравнения. М. Наука. 1971.



## 2. Метод молекулярной динамики

Метод молекулярной динамики применяется для изучения свойств жидкостей, газов и различных поверхностных явлений [Гулд]. Основа этого метода – численное решение классических уравнений Ньютона для рассматриваемой консервативной системы частиц. При этом предполагается выполнение законов сохранения импульса и энергии, причем потенциальная энергия определяется только суммой двухчастичных взаимодействий.

Развитие молекулярной динамики шло двумя путями. Первый, обычно называемый классическим, (когда вычисляются траектории атомов) имеет довольно длительную историю. Он восходит к задаче двухчастичного рассеяния, которая может быть решена аналитически. Позднее классический подход был подкреплен полуклассическими и квантовохимическими расчетами в тех областях, где влияние квантовых эффектов становилось значимым. Вторым путем развития метода молекулярной динамики стало исследование термодинамических и динамических свойств систем. Идеи, лежащие в основу этого пути восходят к работам Ван-дер-Ваальса и Больцмана.

Рассмотрим классическую задачу. Поскольку мы хотим понять качественные свойства системы многих частиц, пойдем на упрощение задачи, предполагая, что динамику можно считать классической, а молекулы – химически инертными шариками. Мы предполагаем также, что сила взаимодействия любых двух молекул зависит только от расстояния между ними. В этом случае полная потенциальная энергия  $U$  определяется суммой двухчастичных взаимодействий:

$$U = V(r_{12}) + V(r_{13}) + \dots + V(r_{23}) + \dots = \sum_{i < j-1}^N V(r_{ij}), \quad (1)$$

где  $V(r_{ij})$  – потенциал двухчастичных взаимодействий, а  $V(r_{ij})$  зависит только от абсолютной величины расстояния  $r_{ij}$  между частицами  $i$  и  $j$ . Парное взаимодействие вида (1) соответствует простым жидкостям, например жидкому аргону. Наиболее важными особенностями  $V(r)$  для простых жидкостей является сильное отталкивание для малых  $r$  и слабое притяжение на больших расстояниях. Одной из наиболее употребительных феноменологических формул для  $V(r)$

является потенциал Леннарда-Джонса (рис. 1):  $V(r) = \left[ \left( \frac{\sigma}{r} \right)^{12} - \left( \frac{\sigma}{r} \right)^6 \right]$ , (2)

где  $\sigma$  – нормированная единица расстояния,  $r$  – расстояние между частицами в единицах  $\sigma$ .

Для каждой из частиц системы выполняются законы Ньютона, закон сохранения импульса, сила взаимодействия любых двух молекул зависит только от расстояния между ними и определяется указанным выше потенциалом взаимодействия. Силы и ускорения определяются следующим образом:

$$F_{i,n+1} = -\text{grad}(U_{i,n}), \quad a_{i,n+1} = \frac{F_{i,n+1}}{m_i}$$

Для расчета траекторий каждой частицы используется алгоритм Верле в скоростной форме.

Расчет координаты:  $x_{n+1} = x_n + v_n \Delta t + a_n (\Delta t)^2$ . (3)

Расчет скорости:  $v_{n+1} = v_n + \frac{1}{2}(a_n + a_{n+1})\Delta t$ . (4)

Расчет ускорения:  $a_{n+1} = \frac{x_{n+1} - 2x_n + x_{n-1}}{(\Delta t)^2}$ . (5)

где  $x_n, v_n, a_n$  – значение координаты, скорости и ускорения на  $n$ -ом шаге расчета,  $x_{n+1}, v_{n+1}, a_{n+1}$  – значение координаты, скорости и ускорения на  $n+1$ -ом шаге расчета,  $\Delta t$  – шаг по времени.

Один из способов более точно промоделировать свойства макроскопической системы проиллюстрирован на рис. 2. Предположим, что частицы 1 и 2 находятся в центральной клетке. Клетка окружена периодически повторяющимися собственными копиями; каждая копия клетки содержит обе частицы в тех же относительных положениях. Когда частица влетает в центральную клетку или вылетает из нее с одной стороны, это перемещение сопровождается одновременным вылетом или влетом копии этой частицы в соседнюю клетку с противоположной стороны. Вследствие использования периодических краевых условий частица 1 взаимодействует с частицей 2 в центральной клетке и со всеми периодическими копиями частицы 2. Однако для короткодействующих взаимодействий мы можем принять правило ближайшей частицы.

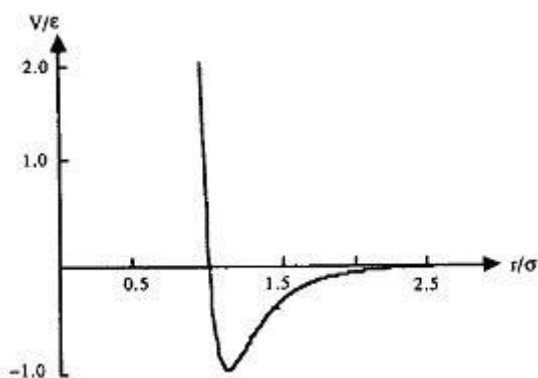


Рис. 1. Потенциал Леннарда-Джонса

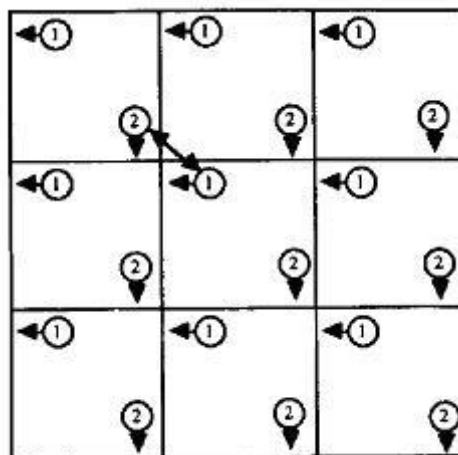


Рис. 2. Пример периодических краевых условий в двумерном случае

**Особенности реализации метода.** Для уменьшения влияния поверхностных эффектов и повышения достоверности результатов используются периодические краевые условия, зависящие от размера моделируемой системы. Однако, в известном алгоритме [1] найдены неточности, не позволяющие применять этот алгоритм в широком диапазоне параметров моделируемой системы. Кроме того, применение известных алгоритмов требует решения следующей проблемы: при большом шаге погрешности интегрирования могут быть значительными, а при малом шаге увеличивается время расчета и время, необходимое для расчёта траекторий молекул.

С учетом этих особенностей на базе алгоритма Верле разработаны и протестированы программные средства для моделирования молекулярных систем с использованием потенциала взаимодействия Леннарда-Джонсона.

Задав начальные скорости частиц (распределение Максвелла) и их координаты, на каждом шаге моделирования используют уравнения равноускоренного движения:

$$x_{i,n+1} = x_{i,n} + v_{i,n} \Delta t + \frac{1}{2} a_{i,n} (\Delta t)^2$$

$$F_{i,n+1} = -grad(U_{i,n})$$

$$a_{i,n+1} = \frac{F_{i,n+1}}{m_i}$$

$$v_{i,n+1} = v_{i,n} + \frac{1}{2} (a_{i,n} + a_{i,n+1})$$

Такой алгоритм называется алгоритмом Верле в скоростной форме.

Алгоритм Верле реализован на различных языках программирования (FORTRAN, Pascal, C++ и др), что позволяет легко интегрировать его в любое приложение, а так же легко модифицировать полученный код. Пример программы моделирования на основе алгоритма Верле, а так же результаты решения простейшей задачи молекулярного моделирования представлены в *Приложениях*.

### **Литература к лекции 2**

1. Гулд Х, Тобочник Я. Компьютерное моделирование в физике: В 2-х частях. Часть 1. М., 1990.
2. Rapaport D. C. The Art of Molecular Dynamics Simulation. Cambridge, 2004.

## II. ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ АНАЛИЗ И МОДЕЛИРОВАНИЕ СЛОЖНЫХ СИСТЕМ

*Цель моделирования – создание осмысленной абстракции реального мира. Такая абстракция должна быть проще самого мира, но при этом отображать его достаточно точно, чтобы модель позволяла предсказывать поведение предметов в реальном мире. Здесь уместно вспомнить школьный глобус.*

### 3. Сложные системы

*Пять признаков сложной системы. Алгоритмическая и объектно-ориентированная декомпозиция. Анализ и способы моделирования сложных систем. Разработка концепции моделей и ее составных частей.*

Любая сложная система характеризуется пятью общими признаками.

1. *"Сложные системы часто являются иерархическими и состоят из взаимозависимых подсистем, которые в свою очередь также могут быть разделены на подсистемы, и т.д., вплоть до самого низкого уровня."*

Отмечают тот факт, что "многие сложные системы имеют почти разложимую иерархическую структуру, является главным фактором, позволяющим нам понять, описать ... такие системы и их части". В самом деле, скорее всего, мы можем понять лишь те системы, которые имеют иерархическую структуру.

Важно осознать, что архитектура сложных систем складывается и из компонентов, и из иерархических отношений этих компонентов. Замечено, что: "все системы имеют подсистемы, и все системы являются частями более крупных систем... Особенности системы обусловлены отношениями между ее частями, а не частями как таковыми". Что же следует считать простейшими элементами системы? Опыт подсказывает нам следующий ответ:

2. *Выбор, какие компоненты в данной системе считаются элементарными, относительно произволен и в большой степени оставляется на усмотрение исследователя.*

Низший уровень для одного наблюдателя может оказаться достаточно высоким для другого.

Иерархические системы называются *разложимыми*, если они могут быть разделены на четко идентифицируемые части, и *почти разложимыми*, если их составляющие не являются абсолютно независимыми. Это подводит нас к следующему общему свойству всех сложных систем:

3. *"Внутрикомпонентная связь обычно сильнее, чем связь между компонентами. Это обстоятельство позволяет отделять "высокочастотные" взаимодействия внутри компонентов от "низкочастотной" динамики взаимодействия между компонентами".*

Это различие внутрикомпонентных и межкомпонентных взаимодействий обуславливает разделение функций между частями системы и дает возможность относительно изолированно изучать каждую часть.

Как мы уже говорили, многие сложные системы организованы достаточно экономными средствами. Существует следующий признак сложных систем:

4. *"Иерархические системы обычно состоят из немногих типов подсистем, по-разному скомбинированных и организованных".*

Иными словам и, разные сложные системы содержат одинаковые структурные части. Эти части могут использовать общие более мелкие компоненты, такие как клетки, или более крупные структуры, типа сосудистых систем, имеющиеся и у растений, и у животных.

Выше отмечалось, что сложные системы имеют тенденцию к развитию во времени. Замечено, что сложные системы будут развиваться из простых гораздо быстрее, если для них существуют устойчивые промежуточные формы. Пятый признак формулируется следующим образом:

5. *"Любая работающая сложная система является результатом развития работавшей более простой системы... Сложная система, спроектированная "с нуля", никогда не заработает. Следует начинать с работающей простой системы".*

В процессе развития системы объекты, первоначально рассматривавшиеся как сложные, становятся элементарными, и из них строятся более сложные системы. Более того, невозможно сразу правильно создать элементарные объекты: с ними надо сначала познакомиться, чтобы больше узнать о реальном поведении системы, и затем уже совершенствовать их.

**Декомпозиция.** При проектировании сложной программной системы необходимо разделять ее на все меньшие и меньшие подсистемы, каждую из которых можно совершенствовать независимо. В этом случае мы не превысим пропускной способности человеческого мозга: для понимания любого уровня системы нам необходимо одновременно держать в уме информацию лишь о немногих ее частях (но не о всех одновременно). Способ управления сложными объектами был известен еще в древности. Он очень прост: «разделяй и властвуй» (*Divide et impera*).

Разбиение системы на составные части называют *декомпозицией*.

Различают алгоритмическую (\*) и объектно-ориентированную (\*\*) декомпозицию.

(\*) В первом случае каждый модуль системы выполняет один из этапов общего процесса. Такой подход применяют при структурном программировании.

(\*\*) Во втором случае система с ее окружением представлены совокупностью автономных действующих лиц, взаимодействующих друг с другом, что позволяет обеспечить более организованное поведение системы.

На вопрос: “существует ли лучший способ декомпозиции сложной системы?” нет однозначного ответа.

**Алгоритмическая декомпозиция.** Большинство из нас формально обучено структурному проектированию "сверху вниз", и мы воспринимаем декомпозицию как обычное разделение алгоритмов, где каждый модуль системы выполняет один из этапов общего процесса. На рис. 2-1 приведен в качестве примера один из продуктов структурного проектирования: структурная схема, которая показывает связи между различными функциональными элементами системы. Данная структурная схема иллюстрирует часть программной схемы, изменяющей содержание управляющего файла. Она была автоматически получена из диаграммы потока данных специальной экспертной системой, которой известны правила структурного проектирования.

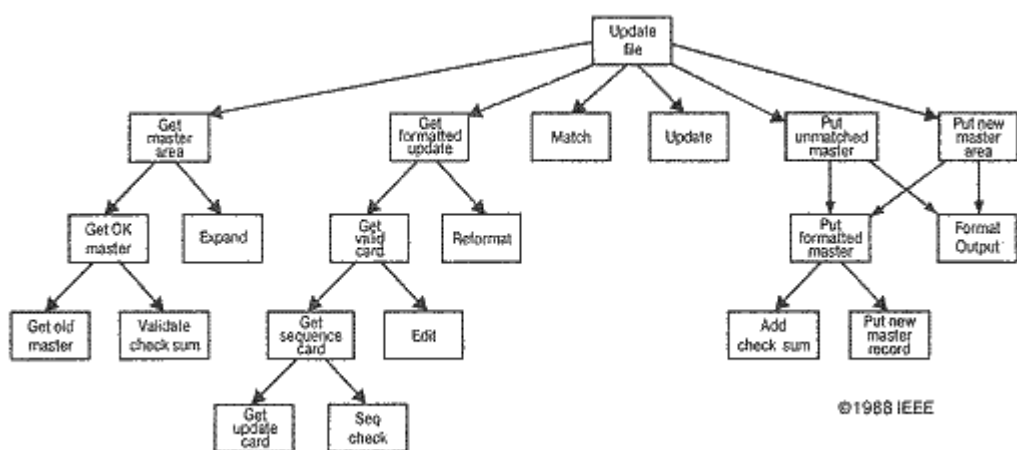


Рис. 2-1. Алгоритмическая декомпозиция.

**Объектно-ориентированная декомпозиция.** Предположим, что у этой задачи существует альтернативный способ декомпозиции. На рис. 2-2 мы разделили систему, выбрав в качестве критерия декомпозиции принадлежность ее элементов к различным абстракциям данной проблемной области. Прежде чем разделять задачу на шаги типа *Get formatted update* (Получить изменения в отформатированном виде) и *Add check sum* (Прибавить к контрольной сумме), мы должны определить такие объекты как *Master File* (Основной файл) и *Check Sum* (Контрольная сумма), которые заимствуются из словаря предметной области.

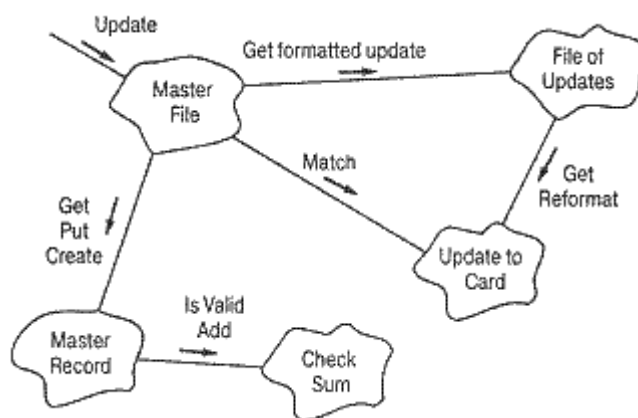


Рис. 2-2. Объектно-ориентированная декомпозиция

Хотя обе схемы решают одну и ту же задачу, но они делают это разными способами. Во второй декомпозиции мир представлен совокупностью автономных действующих лиц, которые взаимодействуют друг с другом, чтобы обеспечить поведение системы, соответствующее более высокому уровню. *Get formatted update* (Получить изменения в отформатированном виде) больше не присутствует в качестве независимого алгоритма; это действие существует теперь как операция над объектом *File of Updates* (Файл изменений). Эта операция создает другой объект - *Update to Card* (Изменения в карте). Таким образом,

каждый объект обладает своим собственным поведением, и каждый из них моделирует некоторый объект реального мира. С этой точки зрения объект является вполне осязаемой вещью, которая демонстрирует вполне определенное поведение. Объекты что-то делают, и мы можем, пошлав им сообщение, попросить их выполнить то-то и то-то. Так как наша декомпозиция основана на объектах, а не на алгоритмах, мы называем ее *объектно-ориентированной декомпозицией*.

**Декомпозиция: алгоритмическая или объектно-ориентированная?** Какая декомпозиция сложной системы правильнее - по алгоритмам или по объектам? В этом вопросе есть подвох, и правильный ответ на него: важны оба аспекта. Разделение по алгоритмам концентрирует внимание на порядке происходящих событий, а разделение по объектам придает особое значение агентам, которые являются либо объектами, либо субъектами действия. Однако мы не можем сконструировать сложную систему одновременно двумя способами, тем более, что эти способы по сути ортогональны. Мы должны начать разделение системы либо по алгоритмам, либо по объектам, а затем, используя полученную структуру, попытаться рассмотреть проблему с другой точки зрения.

Итак, полезнее начинать с объектной декомпозиции. Такое начало поможет нам лучше справиться с внесением организованности в сложные программные системы. Объектная декомпозиция имеет несколько чрезвычайно важных преимуществ перед алгоритмической. Объектная декомпозиция уменьшает размер программных систем за счет повторного использования общих механизмов, что приводит к существенной экономии выразительных средств. Объектно-ориентированные системы более гибки и проще эволюционируют со временем, потому что их схемы базируются на устойчивых промежуточных формах. Действительно, объектная декомпозиция существенно снижает риск при создании сложной программной системы, так как она развивается из меньших систем, в которых мы уже уверены. Более того, объектная декомпозиция помогает нам разобраться в сложной программной системе, предлагая нам разумные решения относительно выбора подпространства большого пространства состояний.

Наряду с декомпозицией при моделировании сложной системы используют принципы *абстракции и иерархии*.

### **Роль абстракции**

В экспериментах Миллера было установлено, что обычно человек может одновременно воспринять лишь  $7 \pm 2$  единицы информации. Это число, по-видимому, не зависит от содержания информации. Как замечает сам Миллер: "Размер нашей памяти накладывает жесткие ограничения на количество информации, которое мы можем воспринять, обработать и запомнить. Организуя поступление входной информации одновременно по нескольким различным каналам и в виде последовательности отдельных порций, мы можем прорвать... этот информационный затор". В современной терминологии это называют разбиением или выделением *абстракций*.

Вулф так описывает этот процесс: "Люди развили чрезвычайно эффективную технологию преодоления сложности. Мы абстрагируемся от нее. Будучи не в состоянии полностью воссоздать сложный объект, мы просто игнорируем не слишком важные детали и, таким образом, имеем дело с обобщенной, идеализированной моделью объекта". Например, изучая процесс

фотосинтеза у растений, мы концентрируем внимание на химических реакциях в определенных клетках листа и не обращаем внимания на остальные части - черенки, жилки и т.д. И хотя мы по-прежнему вынуждены охватывать одновременно значительное количество информации, но благодаря абстракции мы пользуемся единицами информации существенно большего семантического объема. Это особенно верно, когда мы рассматриваем мир с объектно-ориентированной точки зрения, поскольку объекты как абстракции реального мира представляют собой отдельные насыщенные связные информационные единицы.

### **Роль иерархии**

Другим способом, расширяющим информационные единицы, является организация внутри системы иерархий классов и объектов. Объектная структура важна, так как она иллюстрирует схему взаимодействия объектов друг с другом, которое осуществляется с помощью *механизмов* взаимодействия. Структура классов не менее важна: она определяет общность структур и поведения внутри системы. Зачем, например, изучать фотосинтез каждой клетки отдельного листа растения, когда достаточно изучить одну такую клетку, поскольку мы ожидаем, что все остальные ведут себя подобным же образом. И хотя мы рассматриваем каждый объект определенного типа как отдельный, можно предположить, что его поведение будет похоже на поведение других объектов того же типа. Классифицируя объекты по группам родственных абстракций (например, типы клеток растений в противовес клеткам животных), мы четко разделяем общие и уникальные свойства разных объектов, что помогает нам затем справляться со свойственной им сложностью.

Определить роль иерархии в сложной программной системе не всегда легко, так как это требует разработки моделей многих объектов, поведение каждого из которых может отличаться чрезвычайной сложностью. Однако после их определения, структура сложной системы и наше понимание ее сразу во многом проявляются.

**Объектно-ориентированные модели.** Существует ли наилучший метод проектирования? На этот вопрос нет однозначного ответа. По сути дела это завуалированный предыдущий вопрос: "Существует ли лучший способ декомпозиции сложной системы?" Если и существует, то пока он никому не известен. Этот вопрос можно поставить следующим образом: "Как наилучшим способом разделить сложную систему на подсистемы?" Еще раз напомним, что полезнее всего создавать такие модели, которые фокусируют внимание на объектах, найденных в самой предметной области, и образуют то, что мы назвали *объектно-ориентированной декомпозицией*.

*Объектно-ориентированный анализ и проектирование* - это метод, логически приводящий нас к объектно-ориентированной декомпозиции. Применяя объектно-ориентированное проектирование, мы создаем гибкие программы, написанные экономными средствами. При разумном разделении пространства состояний мы добиваемся большей уверенности в правильности нашей программы. В итоге, мы уменьшаем риск при разработке сложных программных систем.

Так как построение моделей крайне важно при проектировании сложных систем, объектно-ориентированное проектирование предлагает богатый выбор моделей, которые представлены на рис. 2-3. Объектно-ориентированные модели



проектирования отражают иерархию и классов, и объектов системы. Эти модели покрывают весь спектр важнейших конструкторских решений, которые необходимо рассматривать при разработке сложной системы, и таким образом помогают создавать проекты, обладающие всеми пятью атрибутами хорошо организованных сложных систем.

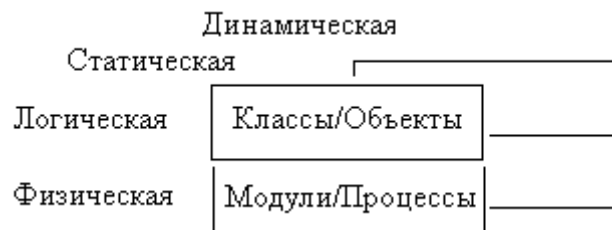


Рис. 2-3. Объектно-ориентированные модели

### Выводы

- Программам присуща сложность, которая нередко превосходит возможности человеческого разума.
- Задача разработчиков программных систем - создать у пользователя разрабатываемой системы иллюзию простоты.
- Сложные структуры часто принимают форму иерархий; полезны обе иерархии: и классов, и объектов.
- Сложные системы обычно создаются на основе устойчивых промежуточных форм.
- Познавательные способности человека ограничены; мы можем раздвинуть их рамки, используя декомпозицию, выделение абстракций и создание иерархий.
- Сложные системы можно исследовать, концентрируя основное внимание либо на объектах, либо на процессах; имеются веские основания использовать объектно-ориентированную декомпозицию, при которой мир рассматривается как упорядоченная совокупность объектов, которые в процессе взаимодействия друг с другом определяют поведение системы.
- Объектно-ориентированный анализ и проектирование - метод, использующий объектную декомпозицию; объектно-ориентированный подход имеет свою систему условных обозначений и предлагает богатый набор логических и физических моделей, с помощью которых мы можем получить представление о различных аспектах.

В качестве упражнения для повторения материала данной лекции предлагается следующая задача.

*Необходимо создать систему подъема на высоту около 20 метров для сбора сока. Система должна быть недорогой и надежной, т.к. пальм очень много (более 10 млн.)*

**Подсказка:** "Любая работающая сложная система является результатом развития работавшей более простой системы... Сложная система, спроектированная "с нуля", никогда не заработает. Следует начинать с работающей простой системы"

### Проектирование сложной системы

Цель проектирования заключается в создании системы, которая:

- "удовлетворяет заданным (возможно, неформальным) функциональным спецификациям;
- согласована с ограничениями, накладываемыми оборудованием;
- удовлетворяет явным и неявным требованиям по эксплуатационным качествам и ресурсопотреблению;
- удовлетворяет явным и неявным критериям дизайна продукта;
- удовлетворяет требованиям к самому процессу разработки, таким, например, как продолжительность и стоимость, а также привлечение дополнительных инструментальных средств".

По предположению Страуструпа: "Цель проектирования - выявление ясной и относительно простой внутренней структуры, иногда называемой *архитектурой*... Проект есть окончательный продукт процесса проектирования".

Под архитектурой понимают структуру классов и объектов вместе.

Проектирование подразумевает учет противоречивых требований. Его продуктами являются модели, позволяющие нам понять структуру будущей системы, сбалансировать требования и наметить схему реализации.

#### 4. Объектная модель

Подход, известный как *объектно-ориентированное программирование* (ООП), позволяет вводить новые типы данных и операции, определяемые пользователем. Он реализуется через объекты и классы.

**Объектно-ориентированное программирование** – это методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

**Объектно-ориентированное проектирование** – это методология проектирования, соединяющая в себе процесс объектной декомпозиции и приемы представления логической и физической, а также статической и динамической моделей проектируемой системы.

**Объектно-ориентированный анализ** – это методология, при которой требования к системе воспринимаются с точки зрения классов и объектов, выявленных в предметной области.

Для документирования анализа и проектирования созданы специальные языки (UML, SDL), а также программные продукты (*Rational Rose*).

#### Элементы объектной модели:

Главные элементы	Дополнительные элементы
- абстрагирование	- типизация
- инкапсуляция	- параллелизм
- модульность	- сохраняемость
- иерархия	

#### **Классы и объекты**

*Природа объекта.* С точки зрения восприятия человека объектом может быть:

- осязаемый и (или) видимый предмет
- нечто воспринимаемое мышлением
- то, на что направлена мысль или действие.

*Объект* обладает *состоянием*, *поведением* и *идентичностью*; структура и поведение схожих объектов определяет общий для них класс. Понятия «объект» и «экземпляр класса» равноценны.

- *-Состояние* – характеризуется перечнем (обычно статическим) всех свойств данного объекта и текущими (обычно динамическими) значениями каждого из этих свойств.  
Пример: записи в БД о сотрудниках. Свойства: возраст, стаж, заработная плата и т.д. Сотрудник Иванов (объект класса) в данный момент времени характеризуется текущими значениями возраста, стажа, заработной платы и т.д.
- *-Поведение* – то, как объект действует и реагирует; это наблюдаемая и проверяемая извне деятельность объекта. Поведение выражается через изменение состояния и передачу сообщений.
- *-Идентичность* - свойство объекта, отличающее его от всех других объектов.

- *Операция* – это услуга, предоставляемая классом своим клиентам. На практике типичный клиент совершает над объектами операции 5 видов. Из них наиболее распространены:
- *Модификатор* – операция, изменяющая состояние объекта.
- *Селектор* – операция, считывающая состояние, но не меняющая состояния объекта.
- *Итератор* – операция, позволяющая организовать доступ ко всем частям объекта в строго определенной последовательности.
- *Класс* – это множество объектов, имеющих общую структуру и общее поведение. Любой конкретный объект является просто экземпляром класса. **Классы необходимы, но недостаточны для декомпозиции сложных систем.**

### *Эволюция объектной модели*

В истории развития программных средств можно выделить несколько поколений языков программирования.

*Языки 1-го поколения* (1954-1958) ориентировались на математические расчеты (FORTRAN I, ALGOL-58).

*Языки 2-го поколения* (1954-1958) позволяли создавать блочные структуры и подпрограммы, работать с файлами (FORTRAN II, ALGOL-60).

*Языки 3-го поколения* (1962-1970) все более широко используют механизм структурирования и (PL/I, ALGOL-68, Pascal, Simula). В Simula использовались классы и абстрактные данные.

*Языки 4-го поколения* – много созданных, но мало выживших.

В начале 70-х годов начал применяться термин «объект» для обозначения того, что может иметь различные проявления, оставаясь целостным.

*Языки C, C++, C#, Java, Object Pascal, PHP*

Общий процесс разработки программной модели системы итеративен:

1. Разработка концепции
2. Анализ (уточнение требований к проекту)
3. Проектирование (формирование требований к проекту)
4. Реализация (код программы)
5. Тестирование (проверка правильности работы программы)
6. Возвращение на один из предшествующих этапов с целью доработки

*Задание для самостоятельной работы: привести примеры классов, объектов и наследования*

### III. ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА МОДЕЛИРОВАНИЯ

Язык моделирования – это набор соглашений относительно принципов предварительного моделирования программы на бумаге. Без такого этапа невозможно создать эффективный программный продукт.

Важное значение имеет выбор программных средств. На начальном этапе целесообразно применять язык моделирования UML, а на этапе реализации – языки C++ или систему Matlab.

Однако, даже при правильном выборе программных средств успех разработки в конечном счете определяется действиями разработчиков, их опытом и умением использовать эти инструментальные средства для достижения цели.

#### 5. Универсальные программные средства

##### Унифицированный язык моделирования UML

Унифицированный язык моделирования UML (Unified Modelling Language) применяется как инструмент объектно-ориентированного анализа и проектирования. Он реализован в программном продукте Rational Rose, позволяющем легко создавать диаграммы классов и объектов.

Рассмотрим основные элементы языка UML, а также основные диаграммы, позволяющие смоделировать программную систему с различных точек зрения (проектировщика, пользователя и т.д.).

Словарь UML включает три вида основных конструкций:

- Сущности – абстракции, являющиеся основными элементами модели;
- Отношения – связи между сущностями
- Диаграммы, группирующие представляющие интерес множества сущностей и отношений.

##### Сущности

В UML имеется четыре типа сущностей:


- Структурные
- Поведенческие
- Группирующие
- Аннотационные


Сущности являются основными объектно-

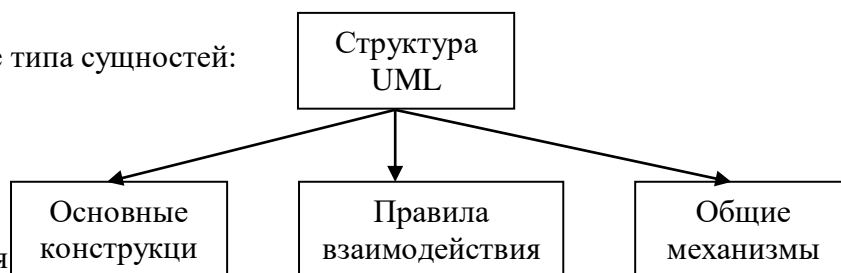
ориентированными элементами языка. С их помощью можно создавать корректные модели.

Структурные сущности


Существует семь разновидностей структурных сущностей, которые, естественно, нашли свое отражение в UML.

 Класс (class) – описание совокупности объектов с общими атрибутами, операциями, отношениями и семантикой


 Интерфейс (interface) – совокупность операций, которые определяют определенную службу (сервис, набор услуг), которые предоставляет класс или компонент. Интерфейс практически никогда не существует сам по себе – обычно он присоединяется к реализующему его классу или компоненту.



Кооперация (collaboration) определяет взаимодействие, она представляет собой совокупность ролей и других элементов, которые, работая вместе, производят некоторый кооперативный эффект, не сводящийся к обычной сумме слагаемых.

 Прецедент (use case) – это описание последовательности выполняемых системой действий, которая производит наблюдаемый результат, значимый для какого-то определенного актера (actor).


Активный класс (active class) – класс, объекты которого вовлечены в один или несколько процессов, или нитей (threads), и поэтому могут инициировать управляющее воздействие.

 Компонент (component) – физически заменяемая часть системы, которая соответствует некоторому набору интерфейсов и обеспечивает его реализацию. Узел (node) – это элемент реальной (физической) системы, который существует во время функционирования программного продукта и представляет собой некоторый вычислительный ресурс, обычно обладающий как минимум некоторым объемом памяти, а часто еще и возможностью обработки.

#### Поведенческие сущности


Поведенческие сущности (behavioral things) являются динамическими составляющими модели UML. Это глаголы языка, они описывают поведение модели во времени и пространстве. Существует два типа поведенческих сущностей:

Взаимодействие (interaction) – это поведение, суть которого заключается в обмене сообщениями (messages) между объектами в рамках конкретного контекста для достижения определенной цели

 Автомат (state machine) – алгоритм поведения, определяющий последовательность состояний, через которые объект или взаимодействие проходят на протяжении своего жизненного цикла в ответ на различные события, а также реакции на эти события


#### Группирующие сущности

Группирующие сущности являются организующими частями модели UML. Это блоки, на которые можно разложить модель. Такая первичная сущность имеется в единственном экземпляре – это пакет.

 Пакеты (packages) представляют собой универсальный механизм организации элементов в группы.

#### Аннотационные сущности


Аннотационные сущности – пояснительные части модели UML. Это комментарии для дополнительного описания, разъяснения или замечания к любому элементу модели. Имеется только один базовый тип аннотационных элементов – примечание.


 Примечание (note) – это просто символ для изображения комментариев или ограничений, присоединенный к элементу или группе элементов.


#### Отношения UML


В языке UML определены четыре типа отношений

- Зависимость
- Ассоциация
- Обобщение
- Реализация

 Зависимость (dependency) – это семантическое отношение между двумя сущностями, при котором изменение одной из них, независимой, может повлиять на семантику другой, зависимой.

 Ассоциация (association) – структурное отношение, описывающее совокупность связей, где под связью понимается некоторая смысловая связь между объектами. Разновидностью ассоциации является агрегирование (aggregation) – так называется структурное отношение между целым и его частями.

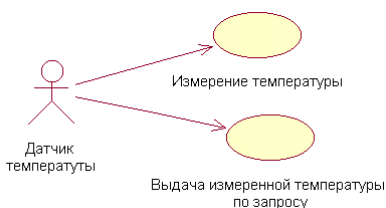
 Обобщение (generalization) – это отношение «специализация\обобщение», при котором объект специализированного элемента (потомок) может быть подставлен вместо объекта обобщенного элемента (родителя, предка). Как и положено в ООП, потомок наследует структуру и поведение своего предка.

 Реализация (realization) – это семантическое отношение между классификаторами, при котором один классификатор определяет обязательство, а другой гарантирует его выполнение. Отношение реализации встречается в двух случаях: во-первых, между интерфейсами и реализующими их классами или компонентами, а во-вторых, между прецедентами и реализующими их кооперациями.

### Диаграммы UML

В распоряжение проектировщика системы Rational Rose предоставляет следующие типы диаграмм, последовательное создание которых позволяет получить полное представление о всей проектируемой системе и об отдельных ее компонентах :

- Use case diagram (диаграммы прецедентов);
- Deployment diagram (диаграммы топологии);
- State Machine diagram
- Statechart diagram (диаграммы состояний);
- Activity diagram (диаграммы активности);
- Interaction diagram (диаграммы взаимодействия);
- Sequence diagram (диаграммы последовательностей действий);
- Collaboration diagram (диаграммы сотрудничества);
- Class diagram (диаграммы классов);
- Component diagram (диаграммы компонент).

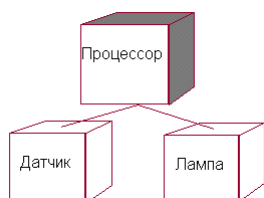


1. Use case diagram (диаграммы прецедентов). Этот вид диаграмм позволяет создать список операций, которые выполняет система. Часто этот вид диаграмм называют диаграммой функций, потому что на основе набора таких диаграмм создается список требований к системе и определяется множество выполняемых системой функций.

Каждая такая диаграмма или, как ее обычно называют, каждый Use case – это описание сценария поведения, которому следуют действующие лица (Actors).

Данный тип диаграмм используется при описании бизнес процессов автоматизируемой предметной области, определении требований к будущей программной системе. Отражает объекты как системы, так и предметной области и задачи, ими выполняемые.

## 2. Deployment diagram (диаграммы топологии)



Этот вид диаграмм предназначен для анализа аппаратной части системы, то есть «железа», а не программ. В прямом переводе с английского Deployment означает «развертывание», но термин «топология» точнее отражает сущность этого типа диаграмм.

Для каждой модели создается только одна такая диаграмма, отображающая процессоры (Processor), устройства (Device) и их соединения. Обычно этот тип диаграмм используется в самом начале проектирования системы для анализа аппаратных средств, на которых она будет эксплуатироваться.

## 3. State Machine diagram (диаграммы состояний)

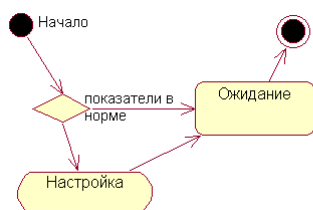
Каждый объект системы, обладающий определенным поведением, может находиться в определенных состояниях, переходить из состояния в состояние, совершая определенные действия в процессе реализации сценария поведения объекта. Поведение большинства объектов реальных систем можно представить с точки зрения теории конечных автоматов, то есть поведение объекта отражается в его состояниях, и данный тип диаграмм позволяет отразить это графически. Для этого используется два вида диаграмм: Statechart diagram (диаграмма состояний) и Activity diagram (диаграмма активности)

### 3.1. Statechart diagram (диаграмма состояний)



Диаграмма состояний (Statechart) предназначена для отображения состояний объектов системы, имеющих сложную модель поведения. Это одна из двух диаграмм State Machine, доступ к которой осуществляется из одного пункта меню.

### 3.2. Activity diagram (диаграммы активности)



Это дальнейшее развитие диаграммы состояний. Фактически данный тип диаграмм может использоваться и для отражения состояний моделируемого объекта, однако, основное назначение Activity diagram в том, чтобы отражать бизнес-процессы объекта.

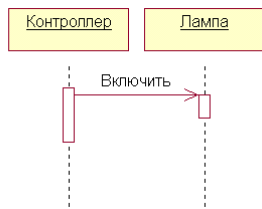
Этот тип диаграмм позволяет показать не только последовательность процессов, но и ветвление и даже синхронизацию процессов. Этот тип диаграмм позволяет проектировать алгоритмы поведения объектов любой сложности, в том числе может использоваться для составления блок-схем.



#### 4. Interaction diagram (диаграммы взаимодействия)

Этот тип диаграмм включает в себя диаграммы Sequence diagram (диаграммы последовательностей действий) и Collaboration diagram (диаграммы сотрудничества). Эти диаграммы позволяют с разных точек зрения рассмотреть взаимодействие объектов в создаваемой системе.

##### 4.1. Sequence diagram (диаграммы последовательностей действий)



Взаимодействие объектов в системе происходит посредством приема и передачи сообщений объектами-клиентами и обработки этих сообщений объектами-серверами. При этом в разных ситуациях одни и те же объекты могут выступать и в качестве клиентов, и в качестве серверов.

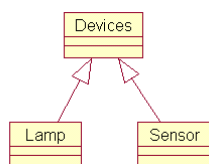
Данный тип диаграмм позволяет отразить последовательность передачи сообщений между объектами. Этот тип диаграммы не акцентирует внимание на конкретном взаимодействии, главный акцент уделяется последовательности приема/передачи сообщений. Для того чтобы окинуть взглядом все взаимосвязи объектов, служит Collaboration diagram.

##### 4.2. Collaboration diagram (диаграммы сотрудничества)



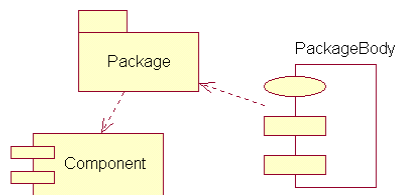
Этот тип диаграмм позволяет описать взаимодействия объектов, абстрагируясь от последовательности передачи сообщений. На этом типе диаграмм в компактном виде отражаются все принимаемые и передаваемые сообщения конкретного объекта и типы этих сообщений.

По причине того, что диаграммы Sequence и Collaboration являются разными взглядами на одни и те же процессы, Rational Rose позволяет создавать из Sequence диаграммы диаграмму Collaboration и наоборот, а также производит автоматическую синхронизацию этих диаграмм.



5. Class diagram (диаграммы классов). Этот тип диаграмм позволяет создавать логическое представление системы, на основе которого создается исходный код описанных классов. Значки диаграммы позволяют отображать сложную иерархию систем, взаимосвязи классов (Classes) и интерфейсов (Interfaces). Данный тип диаграмм противоположен по

содержанию диаграмме Collaboration, на котором отображаются объекты системы. Rational Rose позволяет создавать классы при помощи данного типа диаграмм в различных нотациях. В нотации, предложенной Г. Бучем, которая так и называется Booch, классы изображаются в виде чего-то нечеткого, похожего на облако. Таким образом Г.Буч пытается показать, что класс – это лишь шаблон, по которому в дальнейшем будет создан конкретный объект.



6. Component diagram (диаграммы компонентов). Этот тип диаграмм предназначен для распределения классов и объектов по компонентам при физическом проектировании системы. Часто данный тип диаграмм называют диаграммами модулей.

При проектировании больших систем может оказаться, что система должна быть

разложена на несколько сотен или даже тысяч компонентов, и этот тип диаграмм позволяет не потеряться в обилии модулей и их связей.

**\*) Задание для КСР** – разработать UML - диаграмму с использованием *Rational Rose*

## **6. Специализированные инструментальные средства**

*Граница между универсальными и специализированными средствами условна. В этой лекции рассмотрены особенности разработки объектных моделей в среде MATLAB/Simulink, а также библиотеки для распределенного моделирования (\*).*

В системе MATLAB концепция объектно-ориентированного программирования позволяет реализовывать классы на трех принципах: *сокрытия внутренних данных и методов, переопределения операций и наследования классов.*

Механизм классов в любом языке программирования позволяет создавать новые типы данных. Экземпляры этих новых типов принято называть *объектами классов*. Объекты призваны *скрывать детали реализации* (внутренние данные и методы), а общение с объектами осуществляется через тщательно отобранные открытые методы, составляющие в своей совокупности *интерфейс взаимодействия* с объектами класса. Универсальная теоретическая концепция объектно-ориентированного программирования здесь реализована не в полной мере, так как отсутствуют механизмы, аналогичные виртуальным функциям классов языка C++.

Все основные типы массивов системы MATLAB (числовые и символьные массивы произвольных размерностей, разреженные массивы, структуры и ячейки) представляют собой встроенные классы, а конкретные переменные являются объектами этих классов. Дополнительно имеется возможность вводить новые классы, а также переопределять и доопределять методы всех существующих классов.

Для создания нового класса объектов нужно *спроектировать структуру* системы MATLAB, которая будет хранить данные, принадлежащие объекту, и определить функции-методы работы с этими объектами.

**Язык системы MATLAB** достаточно эффективен. Это наглядный и простой язык *высокого уровня* (очень далекий от машинного языка), похожий во многих отношениях на самый распространенный в мире язык программирования BASIC. Работа с М-функциями в интерпретируемом режиме (без промежуточной стадии полной компиляции в машинный код) создает очень быстро реализуемую цепочку «разработка - запуск – отладка (исправление ошибок) – новый запуск» и т. д. М-язык сверхкомпактен по отношению к массивам и операциям над ними: все математические функции работают с массивами так же легко, как и со скалярными величинами. И, наконец, богатейший набор встроенных математических и графических функций завершают перечисление привлекательных особенностей внутреннего языка пакета MATLAB.

Тем не менее, у любого языка программирования всегда имеются недостатки. Высокоуровневый характер М-языка не позволяет реализовывать алгоритмы работы со сверхсложными структурами данных (деревья и ветвящиеся списки). Интерпретируемый характер этого языка, способствующий скорости разработки, препятствует увеличению быстродействия во время выполнения и т. д.

Код М-функций, сохраняется в *текстовых файлах*, имеющих расширение букву *m*. Именно в таком состоянии эти функции готовы к применению: система MATLAB загружает их в память, преобразовывает в некоторый *промежуточный*

*псевдокод (P-код)*, который уже и выполняется далее в *режиме интерпретации*, когда каждая синтаксически законченная конструкция P-кода заменяется на соответствующий набор машинных инструкций.

**Задание для самостоятельной проработки.** Изучить вопросы:

- как добавлять новые типы данных в системе MATLAB, создавая классы;
- как создавать объекты и управлять ими, если они являются образцами классов MATLAB.

### **Библиотеки для распределенного моделирования (стандарт MPI)**

*MPI - message passing interface* - библиотека функций, предназначенная для поддержки работы параллельных процессов в терминах передачи сообщений.

*Номер процесса* - целое неотрицательное число, являющееся уникальным атрибутом каждого процесса.

*Атрибуты сообщения* - номер процесса-отправителя, номер процесса-получателя и идентификатор сообщения. Для них заведена структура *MPI\_Status*, содержащая три поля: *MPI\_Source* (номер процесса отправителя), *MPI\_Tag* (идентификатор сообщения), *MPI\_Error* (код ошибки); могут быть и добавочные поля.

*Идентификатор сообщения (msgtag)* - атрибут сообщения, являющийся целым неотрицательным числом, лежащим в диапазоне от 0 до 32767. Процессы объединяются в *группы*, могут быть вложенные группы. Внутри группы все процессы перенумерованы. С каждой группой ассоциирован свой *коммуникатор*. Поэтому при осуществлении пересылки необходимо указать идентификатор группы, внутри которой производится эта пересылка. Все процессы содержатся в группе с предопределенным идентификатором *MPI\_COMM\_WORLD*. При описании процедур MPI будем пользоваться словом OUT для обозначения "выходных" параметров, т.е. таких параметров, через которые процедура возвращает результаты.

### **Общие процедуры MPI**

**int MPI\_Init( int\* argc, char\*\*\* argv)**

*MPI\_Init* - инициализация параллельной части приложения. Реальная инициализация для каждого приложения выполняется не более одного раза, а если MPI уже был инициализирован, то никакие действия не выполняются и происходит немедленный возврат из подпрограммы. Все оставшиеся MPI-процедуры могут быть вызваны только после вызова *MPI\_Init*.

Возвращает: в случае успешного выполнения - *MPI\_SUCCESS*, иначе - код ошибки. (То же самое возвращают и все остальные функции, рассматриваемые в данном руководстве.)

**int MPI\_Finalize( void )**

*MPI\_Finalize* - завершение параллельной части приложения. Все последующие обращения к любым MPI-процедурам, в том числе к *MPI\_Init*, запрещены. К моменту вызова *MPI\_Finalize* некоторым процессом все действия, требующие его участия в обмене сообщениями, должны быть завершены. Сложный тип аргументов *MPI\_Init* предусмотрен для того, чтобы передавать всем процессам аргументы *main*:

```
int main(int argc, char** argv)
{
    MPI_Init(&argc, &argv);
    MPI_Finalize();
}
```

#### IV. МОДЕЛИРОВАНИЕ ФОТОФИЗИЧЕСКИХ ПРОЦЕССОВ

##### 7. Ветвящийся процесс (процесс рождения и гибели)

**Ветвящийся процесс (ВП)** – случайный процесс, описывающий широкий круг явлений, связанных с размножением и превращением каких-либо объектов (напр., частиц в физике, молекул в химии, особей какой-либо популяции в биологии и т. п.). Основным математическим предположением, выделяющим класс ветвящихся процессов, является предположение независимости размножения частиц друг от друга. Однородный во времени ветвящийся процесс  $\mu(t)$  с однотипными частицами определяется как марковский процесс со счётным числом состояний  $0, 1, 2, \dots$ , переходные вероятности  $P_{ij}(t)$  которого удовлетворяют дополнительному условию ветвления:

$$P_{ij}(t) = \sum_{k=0}^{\infty} P_{ik}(t) P_{kj}(t) \quad (1)$$

Состояния  $0, 1, 2, \dots$  в ветвящемся процессе (ВП) интерпретируются как числа частиц. Вероятность  $P_{ij}(t)$  равна вероятности

$$P\{\mu(t + t_0) = j | \mu(t_0) = i\}$$

того, что  $i$  частиц за время  $t$  превращаются в  $j$  частиц. Основным аналитическим аппаратом ветвящегося процесса являются производящие функции

$$G(t; s) = \sum_{n=0}^{\infty} P_{0n}(t) s^n \quad (2)$$

Из условия ветвления (1) вытекает равенство

$$G(t + \tau; s) = G(t; G(\tau; s)). \quad (3)$$

В ВП с дискретным временем  $t$  принимает целые неотрицательные значения, и из (3) следует, что  $G(t; s)$  есть  $t$ -кратная итерация функции  $G(s) = G(1; s)$ . Такой процесс иногда называют процессом Гальтона-Ватсона. В ветвящемся процессе с непрерывным временем предполагается, что  $t \in [0, \infty)$ , и существует правая производная

$$\left. \frac{\partial G(t; s)}{\partial t} \right|_{t=0} = g(s).$$

Из (3) следует, что  $G(t; s)$  удовлетворяет дифференциальному уравнению

$$\frac{\partial G(t; s)}{\partial t} = g(G(t; s)) \quad (4)$$

и начальному условию  $G(0; s) = s$ .

С помощью производящих функций рассчитаем вероятность наблюдения  $n$  электронов на  $k$ -той ступени электронного умножителя.

Для одного каскада:

$$P_1(n) = \frac{m^n e^{-m}}{n!}, \quad (*)$$

$m$  – усиление каскада.

Производящая функция

$$G_1(t; s) = \sum_{n=0}^{\infty} P_{1n}(t) s^n$$

Для  $k$  каскадов производящая функция может быть выражена: [1,3]

$$G_k(s) = \sum_{n=0}^{\infty} P_k(n) s^n \quad (2)$$

Заметим, что  $G_k(s)$  выражено как ряд Маклорена от  $P_k(n)$ , поэтому коэффициенты при  $s^n$  будут

$$P_k(n) = \frac{d^n G_k(s)}{ds^n} \Big|_{s=0} \quad (3)$$

Используя записанные выше выражения

$$G_k(s) = \sum_{n=0}^{\infty} P_k(n) s^n \quad (4)$$

при  $n \neq 0$  и формулу Лейбница

$$\frac{d^n}{ds^n} (f(s)g(s)) = \sum_{i=0}^n \binom{n}{i} \frac{d^i f(s)}{ds^i} \frac{d^{n-i} g(s)}{ds^{n-i}} \quad (5)$$

Для  $n \neq 0$  получим после подстановки:

$$P_k(n) = \sum_{i=0}^n \binom{n}{i} P_k(i) \frac{d^{n-i} G_k(s)}{ds^{n-i}} \Big|_{s=0} \quad (6)$$

Окончательное итерационное выражение получается заменой  $G_k^{(i)}(s)$  и  $G_{k-1}^{(n-i)}(s)$ :

$$P_k(n) = \sum_{i=0}^n \binom{n}{i} P_k(i) G_{k-1}^{(n-i)}(s) \Big|_{s=0}, \quad n \neq 0, k \geq 1 \quad (7)$$

Случай  $n=0$  можно рассмотреть отдельно. Из (4) получаем:

$$P_k(0) = G_k(0)$$

Следовательно

$$P_k(n) = \sum_{i=0}^n \binom{n}{i} P_k(i) G_{k-1}^{(n-i)}(s) \Big|_{s=0}, \quad n \geq 0, k \geq 1 \quad (8)$$

$P_k(n)$  – распределение вероятности наблюдать  $n$  электронов на  $k$ -й ступени умножения (каскаде). Итак, итерационные выражения (7) и (8) позволяют определить вероятности  $P_k(n)$  для  $k > 1$ .

ВП применяются в расчетах многих реальных биологических, генетических, физических, химических или технических процессов. В реальных процессах часто нарушается условие независимости размножения различных частиц и, наоборот, при размножении имеется взаимодействие частиц. Так обстоит дело во многих биологических процессах размножения, в процессах распространения эпидемий, в бимолекулярных химических реакциях и т. п. Однако начальные стадии развития таких процессов можно рассчитывать с помощью соответственно подобранных моделей ВП. Это делается в тех случаях, когда в среде имеется не очень много активных частиц, которые при малых концентрациях почти не встречаются друг с другом, а изменения состояния системы происходят при встречах этих активных частиц с частицами среды. В процессах эпидемии, например, такими «активными частицами» можно считать больных индивидуумов. В генетике с помощью ВП можно рассчитывать явления, связанные с мутациями. ВП с конечным числом типов частиц может служить моделью при расчетах ценных реакций; ветвящийся процесс с диффузией частиц — моделью нейтронных процессов в ядерных реакторах. Явления, возникающие в ливнях космических лучей, также могут изучаться с помощью ветвящихся процессов. В телефонии расчет некоторых систем с ожиданием сводится к моделям ВП.

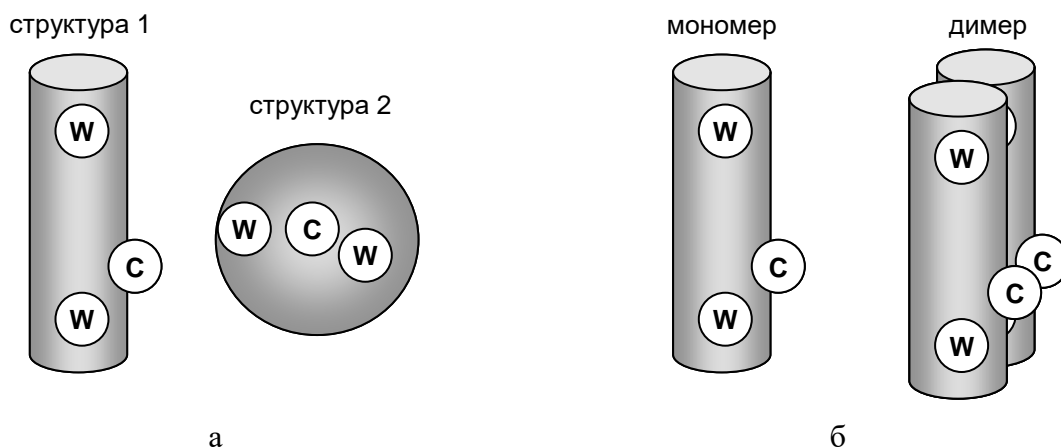
## 8. Имитационное моделирование дискретных процессов

**1. Введение.** Лекция будет построена по следующей схеме. Вначале выясним, для каких целей используют флуоресценцию в молекулярной биологии и физике и для чего нам нужно уметь моделировать фотофизические процессы. Затем, выбрав простейшую систему, мы построим ее аналитическую и имитационную модели. После этого будем вносить усложнения в рассматриваемую систему, постепенно усложняя и модели.

Рассмотрим следующую задачу. Пусть у нас имеется некоторая молекула, например протеин (белок) неизвестной формы. Нам известно, что в этом протеине несколько аминокислот обладающих флуоресцентными свойствами (например – триптофаны) а также есть цистеин, на который искусственно можно поместить флуорофор<sup>1</sup> (Флуорофор – молекула или ее часть обладающая флуоресцентными свойствами). Задачи исследователя:

- 1) определить структуру протеина (или выбрать из ряда структур полученных с помощью молекулярного моделирования);
- 2) получить информацию о агрегированности протеинов – представлена ли популяция мономерами, димерами или полимерами и в каком соотношении (рис. 1).

На эти вопросы можно ответить, исследовав систему флуоресцентными методами. В частности информация может быть получена из спектров флуоресценции молекул (полярность раствора и окружение влияет на эмиссионный спектр), временных характеристик затухания флуоресценции (скорость затухания зависит от происходящих процессов) и т.д.



**Рисунок 1.** Пояснение к постановке задачи: определение структуры протеина (а) и агрегирования (б). На рисунке показаны положения аминокислоты триптофана (W) и цистеина (C).

Пока речь идет о качественном изучении системы (ответы на поставленные вопросы при качественном анализе будут да/нет/не ясно) – можно обойтись без моделирования, "на глаз" оценивая экспериментальные результаты. Как только речь заходит о количественном анализе – приходится строить модели. Для простых систем достаточно просто записать аналитические выражения, описывающие их поведение. Для более сложных систем единственным

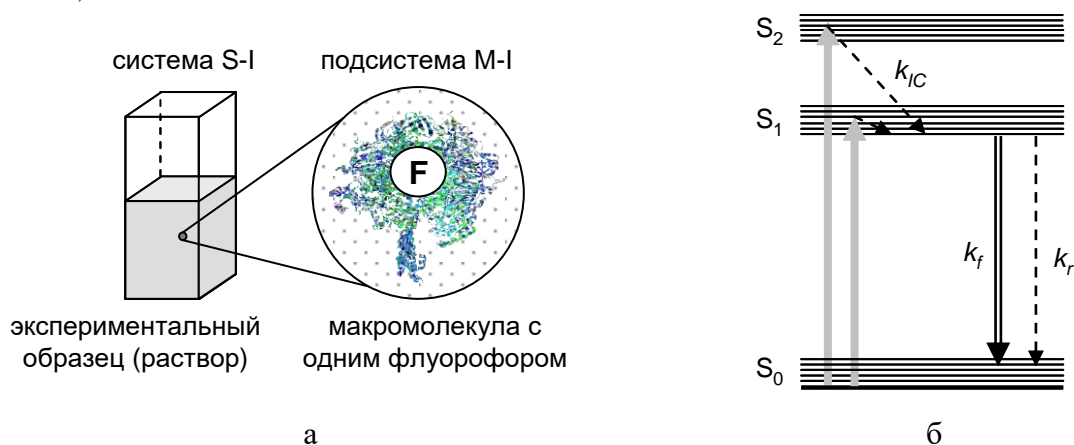
адекватным способом моделирования на сегодняшний день остается имитационное моделирование.

## 2. Основы флуоресцентных методов: однотипные флуорофоры

Мы начнем с простейшего случая. Опишем физику происходящих процессов и методы их моделирования. Потом начнем усложнять систему и вводить в моделирование все новые элементы.

**Физическая система.** Пусть имеется некоторый раствор (система S-I) состоящий из большого числа идентичных молекулярных образований (систем M-I). Каждая система M-I включает 1 флуорофор (рис. 2а). Схема энергетических уровней представлена на рис. 2б. Следует обратить внимание, что при флуоресценции (времена жизни  $\sim ns$ ) осуществляются переходы только между синглетными состояниями. Если в системе есть возможность возбуждения триплетного состояния, это ведет к фосфоресценции (времена жизни  $\sim 10^{-2}-10^2$  с) при переходе с возбужденного триплетного уровня на основной  $S_0$ , т.к. такие переходы являются запрещенными.

Исходя из этой схемы возбуждение возможно на 2х длинах волн (спектр поглощения будет иметь 2 пика), и испускание на одной длине волны (переход  $S_1 \rightarrow S_0$ ).



**Рисунок 2.** Простейшая экспериментальная система S-I (раствор в кювете) состоящая из большого числа эквивалентных подсистем M-I (а) и схема энергетических уровней флуорофора (б), где  $k_{IC}$  – скорость внутреннего преобразования энергии,  $k_f$  и  $k_r$  – скорости излучательной (флуоресценция) и безизлучательной (обычно – нагрев) дезактивации.

Будем считать что переход  $S_2 \rightarrow S_1$  осуществляется без испускания фотона. Так называемое внутренне преобразование  $S_2 \rightarrow S_1$ , как правило, достаточно быстрый процесс, при этом временем жизни на уровне  $S_2$  ( $\sim 10^{-15}$ с) можно пренебречь. Испускание фотона происходит с нижнего подуровня  $S_1$  на один из подуровней основного состояния  $S_0$  со скоростью  $k_f$ .

Флуоресцентные свойства систем M-I и S-I можно описать рядом параметров, которые мы в последствие будем использовать при моделировании. Перечислим их.

- Длина волны возбуждения системы. Как правило, возбуждают систему на одной длине волны, например соответствующей переходу  $S_0 \rightarrow S_1$ . Обозначим эту длину волны  $\lambda_{ex}$  и соответствующую ей частоту –  $\nu_{ex} \sim 1/\lambda_{ex}$ .

- Коэффициент экстинкции  $\varepsilon(\lambda)$  (molar extinction coefficient) характеризует эффективность поглощения флуорофором излучения на длине волны  $\lambda$  (в данном случае – с последующим переходом в возбужденное состояние). Измеряется в  $\text{см} \cdot (\text{моль/литр})^{-1}$  или  $\text{м}^2 \cdot \text{моль}^{-1}$  в СИ.
- Скорости излучательной и безизлучательной дезактивации  $k_f$  и  $k_r$  соответственно. Обозначим суммарную скорость дезактивации  $k_0 = k_f + k_r$ .
- Время жизни в возбужденном состоянии  $S_1$  – зависящая величина, равная по определению  $\tau_0 = k_0^{-1}$ .
- Квантовый выход флуорофора  $q$  – вероятность испускания энергии в виде фотона  $q = k_f / (k_f + k_r)$ .
- Длина волны максимума флуоресценции  $\lambda_{em} \sim 1/\nu_{em}$ . Обычно детектируют излучение на длине волны, близкой к  $\lambda_{em}$ .

**Аналитическая модель затухания флуоресценции.** Теперь перейдем к описанию эксперимента (сначала – в упрощенной форме). Итак, допустим, что в момент времени  $t=0$  мы на мгновение осветили систему  $\delta$ -импульсом лазера на длине волны  $\lambda_{ex}$ . При этом  $N_0$  молекул системы S-I перешли в возбужденное состояние.

Скорость перехода  $N$  молекул в основное состояние в момент времени  $t$  по определению равна:

$$\frac{dN}{dt} = -k_0 N \quad (1)$$

Это – дифференциальное уравнение первого порядка разрешенное относительно производной. Вместе с начальными условиями  $N(t=0)=N_0$  оно представляет собой задачу Коши. Решаем с помощью разделения переменных и интегрирования:

$$\int \frac{dN}{N(t)} = - \int k_0 dt$$

$$\ln N(t) = c - k_0 t \quad (2)$$

$$N(t) = N_0 e^{-k_0 t}$$

Поскольку интенсивность флуоресценции в течение некоторого малого времени пропорциональна величине молекул находящихся в возбужденном состоянии, можем записать аналитический закон флуоресценции:

$$f(t) = F_0 e^{-k_0 t}, \quad (3)$$

где  $F_0$  – значение флуоресценции в  $t=0$ . При желании это значение можно связать с концентрацией флуорофоров в системе  $C$ , коэффициентом экстинкции и квантовым выходом.

$$F_0 = C \varepsilon q \quad (4)$$

где  $\gamma$  – некоторая аппаратная константа, характеризующая мощность исходного импульса, эффективность детектора и прочее.

**Аналитическая модель спектра флуоресценции.** Как правило, в гомогенной системе (все флуорофоры в одинаковых условиях) при наличии одного перехода уширение молекулярной спектральной линии – гауссово. Т.е. энергия, испускающаяся при переходе  $S_1 \rightarrow S_0$ , является нормально распределенной



случайной величиной с математическим ожиданием  $\lambda_{em}$ , и дисперсией  $\sigma$ , зависящей от ширины уровней. Обозначая энергию испускания  $\Delta E$ , и вспомнив, что  $\Delta E \sim \nu$  (частоте) можно аналитически записать закон флуоресценции в этом случае. Для спектрального представления будем обозначать флуоресценцию функцией  $s(\nu)$  или  $s(\lambda)$ .

$$s(\nu) = \frac{1}{\sigma} \exp\left[-\frac{(\nu - \lambda_{em})^2}{\sigma^2}\right] \quad (5)$$

Считая, что  $\nu$  задана в  $\text{см}^{-1}$ , а  $\lambda$  – в нм, выражение 4 можно переписать для случая экспериментально определяемого спектра (спектрометры, как правило, измеряют в нм с постоянным шагом).

$$s(\lambda) = \frac{1}{\sigma} \exp\left[-\frac{(\lambda - \lambda_{em})^2}{\sigma^2}\right] \quad (6)$$

Построенный график  $s(\lambda)$ , будет несимметричным, более пологим в длинноволновой области.

**Имитационная модель.** Посмотрим на аналитическую функцию (3). Она представляет собой ненормированную плотность вероятности времен испускания фотонов возбужденной системой. Значит для имитационного моделирования систем (и этой и более сложной) надо научиться моделировать моменты наступления событий, распределенные по этому закону. Воспользуемся методом обратных функций.

Посчитаем функцию распределения времен испускания  $G(t)$  на основании плотности вероятности  $g(t)$  (полученной из (3) нормированием площади под кривой на 1).

$$G(t) = \int_0^t g(t') dt' = \int_0^t k e^{-kt'} dt' = 1 - e^{-kt} \quad (7)$$

Сгенерировав теперь базовую случайную величину  $\alpha$ , распределенную равномерно на интервале (0,1) и приравняв ее  $G(t)$  решим полученное уравнение относительно  $t$ .

$$\begin{aligned} 1 - e^{-kt} &= \alpha \\ -kt &= \ln(1 - \alpha) \\ t &= -k^{-1} \ln(1 - \alpha) \end{aligned} \quad (8)$$

Поскольку случайные величины  $\alpha$  и  $1 - \alpha$  равноправны, конечная формула

$$t = -k^{-1} \ln(\alpha) \quad (9)$$

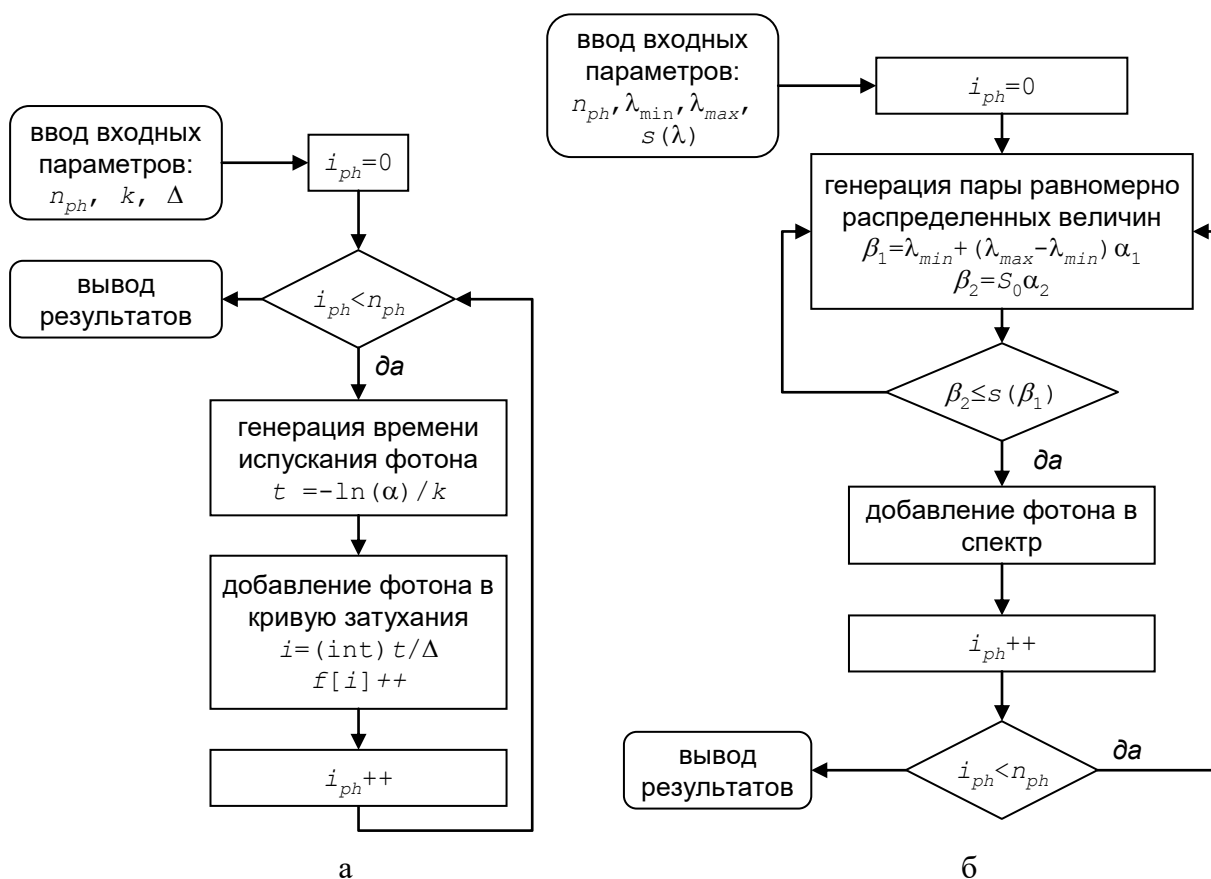
При имитационном моделировании отслеживают поведение единичного фотона в системе, поэтому вводится еще один параметр – число зафиксированных фотонов  $n_{ph}$ . Для большого числа фотонов строится гистограмма времен

испускания, которая аналогична экспериментальным данным – кривым затухания флуоресценции. Алгоритм моделирования представлен ниже.

Иногда требуется смоделировать не только время испускания фотона, но и его энергию. Для этого воспользуемся методом Неймана. В качестве функции описывающей поведение спектра флуоресценции можно брать как экспериментальные данные, так и аналитическую функцию (6). Пусть спектр задан в области от  $\lambda_{min}$  до  $\lambda_{max}$  и изменяется от 0 до  $S_0$ . В методе Неймана генерируется пара равномерно распределенных случайных величин

$$\begin{aligned} \beta_1 &\in (\lambda_{min}, \lambda_{max}) \\ \beta_2 &\in (0, S_0) \end{aligned} \quad (10)$$

и делается проверка, попадает ли точка с координатами  $(\beta_1, \beta_2)$  под кривую  $s(\lambda)$ , т.е. выполняется ли условие  $\beta_2 \leq s(\beta_1)$ . Если условие выполняется – точка принимается и заносится в гистограмму спектра.



**Рисунок 3.** Блок схема имитационного моделирования системы S-I для получения кривых затухания (а) и спектров (б). Генератор базовой случайной величины обозначен символом  $\alpha$

### 3. Система нескольких типов не взаимодействующих флуорофоров

Рассмотрим ситуацию когда в системе (назовем ее S-II) есть несколько типов ( $n$ ) не взаимодействующих флуорофоров с ниже перечисленными параметрами. Параметры системы:

- $k_1, k_2$  – скорости релаксации;
- $q_1, q_2$  – квантовые выходы флуорофоров;
- $\varepsilon_1, \varepsilon_2$  – коэффициенты экстинкции на длине волны  $\lambda_{ex}$ ;
- $s_1(\lambda), s_2(\lambda)$  – нормированные по площади (площадь под спектром = 1) спектры флуоресценции.
- $C_1, C_2$  – концентрации (или количества молекул) флуорофоров.

**Аналитическая модель.** Опуская непосредственный вывод выражения (см. вывод (3) и (4)), запишем аналитический вид флуоресценции системы S-II.

$$I(\lambda) = I_1(\lambda) + I_2(\lambda) \quad (11)$$

Определим, чему пропорционально значение  $F_i$ :  $C_i, \varepsilon_i, q_i, s_i(\lambda_{det})$ , где  $\lambda_{det}$  – длина волны детектирования. Таким образом

$$I(\lambda) = \sum_i C_i \varepsilon_i q_i s_i(\lambda_{det}) \quad (12)$$

**Имитационная модель.** Теперь рассмотрим имитационное моделирование флуоресценции такой системы. Последовательность действий при этом следующая.

1) Следует разыграть, флуорофор какого типа возбудится прилетевшим фотоном. Вероятность возбуждения флуорофоров типа  $i$ :

$$p_i = \frac{\varepsilon_i C_i}{\sum_{j=1}^n \varepsilon_j C_j} \quad (13)$$

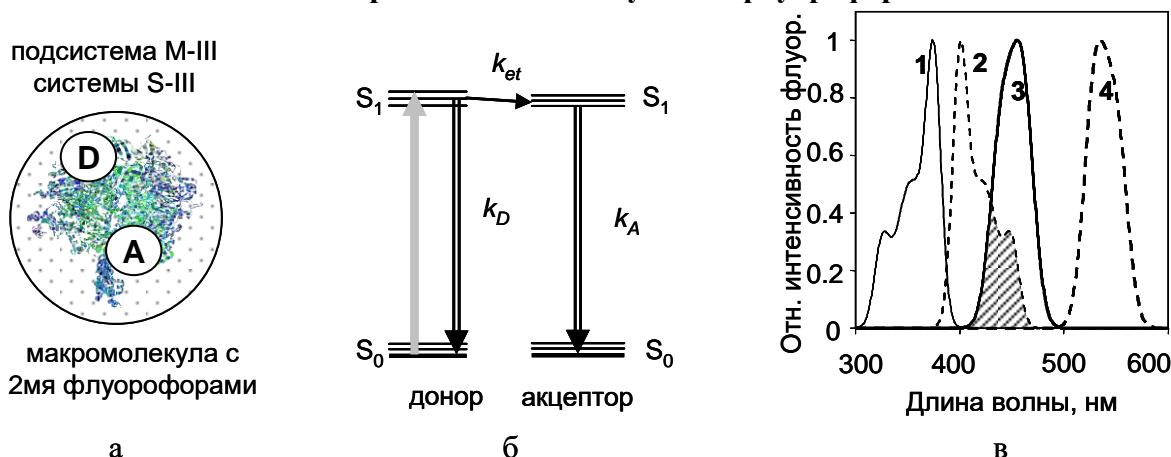
Для выбора события генерируется базовая случайная величина  $\alpha$  и проверяется, в какой карман гистограммы с границами  $[0, p_1, p_1+p_2, \dots, \sum_{i=1}^{n-1} p_i, 1]$  она попадает. Попадание в  $i$ -й карман означает что возбуждается один из флуорофоров  $i$ -го типа.

Затем по формуле (9) генерируется время нахождения возбуждения на флуорофоре. Прежде чем заносить фотон в гистограмму флуоресценции следует разыграть вероятность детектирования, поскольку для флуорофоров разных типов она будет разной. Вероятность детектирования для  $i$ -го флуорофора пропорциональна квантовому выходу и значению спектра на длине волны детектирования  $\lambda_{det}$

$$d_i = \frac{q_i f_i(\lambda_{det})}{\sum_{j=1}^n q_j f_j(\lambda_{det})} \quad (14)$$

Для оптимизации алгоритма формулы (13) и (14) можно объединить и вычислять вероятность пропорциональную произведению  $C_i \varepsilon_i q_i s_i(\lambda_{det})$ .

#### 4. Система с парой взаимодействующих флуорофоров



**Рисунок 4.** Молекула с 2мя флуорофорами (а). Схема уровней (б) при резонансном переносе энергии с молекулы D (донор) на A (акцептор),  $k_D$  и  $k_A$  – скорости релаксации,  $k_{et}$  – скорость переноса энергии. На рисунке (в) представлен пример спектров возбуждения (1-доноров, 3-акцепторов) и испускания (2-доноров, 4- акцепторов).

Рассмотрим теперь систему S-III, в которой макромолекулы (M-III) содержат по 2 взаимодействующих флуорофора (рис. 4). Возбуждение поглощается донором (D) и с некоторой вероятностью передается на акцептор (A). В случае диполь-дипольного кулоновского взаимодействия флуорофоров имеем для скорости переноса энергии в уединенной донор-акцепторной паре следующее выражение

$$k_{et} = k_D \left( \frac{R_0}{r} \right)^6 \quad (15)$$

Здесь донор-акцепторное расстояние  $r$  постоянно в течение времени жизни донора в возбужденном состоянии,  $k_D$  – скорость релаксации донора в отсутствие акцептора,  $R_0$  – так называемое расстояние Ферстера, характеризующее перенос энергии с донора на акцептор с вероятностью 0.5, т.е. когда  $k_{et} = k_D$ . Этот параметр вычисляется исходя из спектроскопических данных для используемых в эксперименте флуорофоров.

**Аналитическая модель.** Используя схему аналогичную выводу выражений (3) и (4) можно определить, что аналитический вид флуоресценции системы S-III

$$f_{ID} = I_0 \left( \frac{k_{et}}{k_D + k_{et}} \right) \quad (16)$$

При желании, решив систему дифференциальных уравнений можно получить и выражение для флуоресценции акцептора.

$$f_{IA} = I_0 \left( \frac{k_{et}}{k_A + k_{et}} \right) \quad (17)$$

Вероятность переноса энергии, называемая иначе эффективностью переноса энергии вычисляется из скоростей релаксации и переноса:

$$F = \frac{k_{et}}{k_{et} + k_D} \approx \frac{1}{1 + k_D/k_{et}} \quad (18)$$

В случае, когда  $r$  меняется от молекулы к молекуле для адекватного аналитического описания необходимо учитывать функцию распределения расстояний, которая, как правило, не известна.

**Имитационная модель.** Теперь рассмотрим имитационное моделирование флуоресценции такой системы. Будем считать, что возбуждение системы происходит на длине волны поглощения доноров, т.е. акцепторы исходно не возбуждаются. Для полноты картины будем моделировать испускание как донора, так и акцептора.

Последовательность действий при этом следующая.

1) Разыгрывается время нахождения возбуждения на доноре (поскольку все доноры одинаковы – считаем, что донор возбуждается со 100% вероятностью). При вычислении этого времени следует использовать полную скорость дезактивации донора ( $k_D + k_{et}$ ).

$$t_D = (k_D + k_{et})^{-1} \ln \xi \quad (19)$$

2) С учетом значений скорости переноса и релаксации донора разыгрывается, какое событие произойдет с возбуждением донора – испустится или же перейдет на акцептор. Соответствующие вероятности пропорциональны скоростям.

3) Если испускается фотон, добавляется единица в гистограмму флуоресценции донора с временем испускания  $t_D$ . В противном случае разыгрывается время испускания с акцептора по формуле

$$t_A = t_D - k_A^{-1} \ln \eta \quad (20)$$

В случае, когда молекулярные системы М-III разнородны, в алгоритм добавляется разыгрывание того, донор какой именно из подсистем М-III возбуждается.

## 5. Верификация модели

В соответствии со схемой (рис. 5) построенную модель необходимо проверить. Проверка осуществляется, как правило, сравнением результатов аналитического и имитационного моделирований для простейших случаев.



Рисунок 5. Общая схема вычислительного эксперимента

Для оценки качества моделирования могут быть использованы различные статистические критерии. Наиболее используемые: критерий хи-квадрат, график взвешенных остатков и автокорреляционная функция. В настоящей работе были использованы критерий хи-квадрат и взвешенные остатки.

**Критерий хи-квадрат.** Данный критерий представляет собой меру отклонения экспериментальных данных от теоретических и может быть представлен следующим соотношением:

$$\chi^2 = \frac{1}{N} \sum_{k=1}^m \frac{(f_k - F_k)^2}{F_k} \quad (21)$$

где  $N$  – число детектированных фотонов;

$m$  – количество интервалов разбиения гистограммы;

$I_k$  – число событий, попавших в  $k$ -ый карман диаграммы при имитационном моделировании;

$f_k$  – аналитическое предсказание числа событий, попавших в  $k$ -ый интервал разбиения.

Статистическая интерпретация критерия следующая: значение  $\chi^2 < 0.8$  указывает на слишком малый объем данных;  $\chi^2 > 1.2$  свидетельствует о значительном расхождении анализируемых теоретических и экспериментальных данных.

**График взвешенных остатков.** Взвешенные остатки вычисляются для каждого интервала разбиения по следующей формуле:

$$\rho_k = \frac{I_k - f_k}{\sqrt{(I_k + f_k)/2}} \quad (22)$$

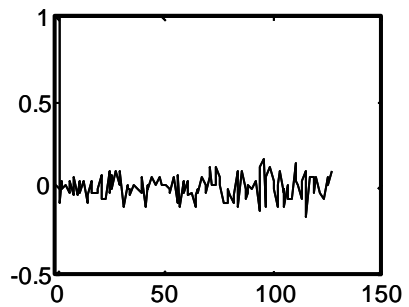
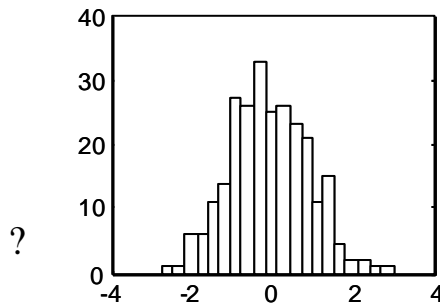
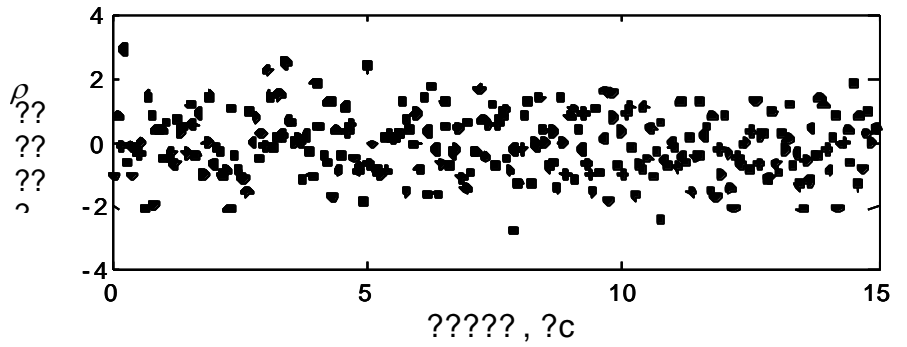
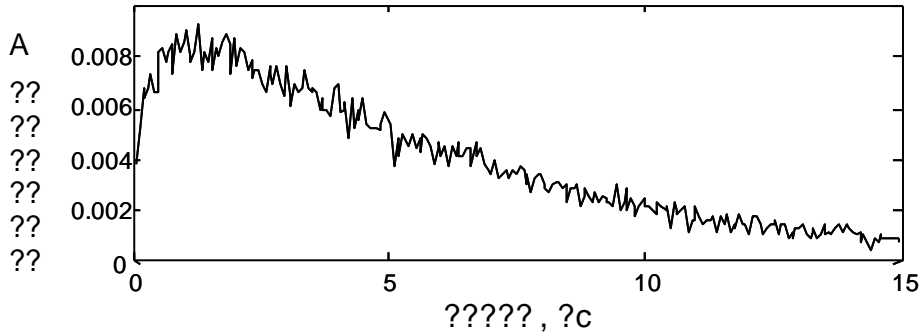
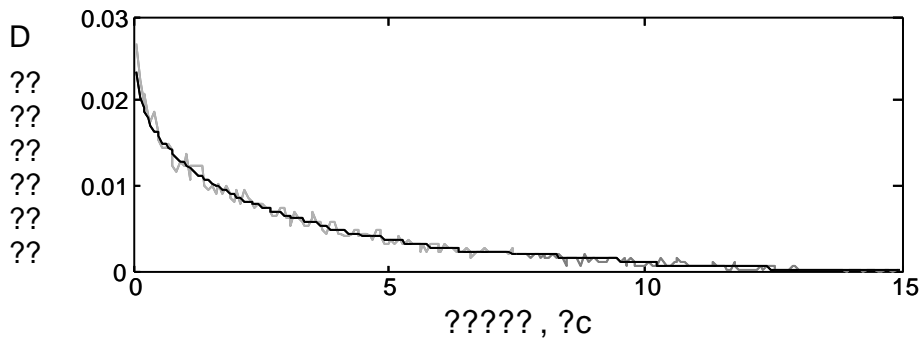
Отложив по оси абсцисс время  $t$ , а по оси ординат  $\rho_k$ , получим график взвешенных остатков. При хорошем совпадении смоделированных данных с теоретическими остатки представляют из себя нормально распределённую случайную величину с нулевым математическим ожиданием.

**Автокорреляционная функция взвешенных остатков.** Автокорреляционная функция характеризует линейную взаимосвязь между отсчетами взвешенных остатков и вычисляется по следующей формуле (не нормирована):

$$c_j = \sum_{i=1}^n \rho_i \rho_{i-j} \quad (23)$$

При хорошем совпадении автокорреляционная функция должна быть близка к дельта функции, а шум ее должен быть распределен около нуля.

На рис. 6 представлен пример результата верификации имитационной модели испускания донора в окружении акцепторов (проверка модели флуоресценции акцепторов не проводилась из-за сложности получения аналитических выражений).



?????? 6 ????? ?????????? ????? ?????????? ?????????? ?????????? .  $\chi^2=0.997$   
 ????? (???? ???? ) ?????????????? ?????????????? (???? ???? ) ??????????????  
 ????? (?). ?????????????? ?????????????? (?). ?????????????? ?????????????? (?). ??????????????  
 ?????????????? (?). ?????????????????????? ?????????? ?????????? (?). ??????????????



## V. ПРИМЕРЫ МОДЕЛЕЙ СИСТЕМ

### 9. Модели электронных приборов

**Принципы моделирования электронных приборов.** Существует два основных подхода к моделированию характеристик электронных приборов, один из которых условно можно назвать *эмпирическим*, другой – *физическим*. В первом случае для описания рабочих характеристик прибора используют экспериментальные зависимости токов или напряжений, таблицы, эквивалентные схемы. Во втором случае исходят из физических основ.

Особенность физического подхода заключается в том, что рабочая область прибора разбивается на ряд элементарных объемов, связанных друг с другом. В каждом объеме рассматривают набор определенных физических процессов. Такие модели наилучшим образом можно использовать для анализа и оптимизации характеристик разрабатываемых приборов. Основная проблема, которую необходимо решить в этом случае, заключается в определении наиболее существенных процессов, которые определяют исследуемые характеристики данного прибора. На практике применяют модели, сочетающие в себе элементы различных подходов. Например, модель Эберса-Молла построена полумэмпирическим путем.

*Вакуумные приборы. Сходство и отличия ФЭУ и (МКП).*

В том и другом случае для усиления используется вторично-электронная эмиссия. Но если число каскадов умножения для ФЭУ фиксировано, то в случае МКП для каждой лавины оно меняется случайным образом. Следовательно, МКП наследует многие из свойств ФЭУ.

Создание достаточно полной модели прибора не всегда возможно, поэтому обычно ограничиваются приемлемым уровнем детализации. Поясним это на примере ФЭУ и МКП

<9.2> Модель МКП.

Микроканальная пластина (МКП) состоит из большого числа параллельных каналов, внутренняя поверхность которых покрыта полупроводниковым слоем с определенной проводимостью и коэффициентом вторичной электронной эмиссии  $\delta > 1$ . Простейшая «шаговая» модель МКП предполагает постоянство угла отражения электронов по всей длине канала и отсутствие разброса энергии вторичных электронов.

Рассматривая движение электрона в канале, легко видеть, что время его пролета от точки вылета до удара о стенку канала определяется проекцией вектора начальной скорости на плоскость поперечного сечения канала в месте вылета. Среднее время

$$t_e = \frac{d_k}{\sqrt{\frac{e U_b}{m}}} \sqrt{\frac{6}{\pi}} = \frac{d_k}{\sqrt{\frac{e U_b^*}{m}}}$$

т. е. определяется не средней начальной энергией  $\bar{u}_0$ , а несколько меньшим значением энергии электронов  $\bar{u}_0^* = \frac{\pi}{6} \bar{u}_0$ , так что при  $\bar{u}_0 = 2 \text{ эВ}$  величина  $\bar{u}_0^*$  может быть принята равной 1 эВ.

Все это время  $\bar{t}$  электрон находится под действием сил ускоряющего электрического поля напряженностью  $E = \frac{U_{\text{МКП}}}{l_k}$ , где  $U_{\text{МКП}}$  – напряжение на МКП,  $l_k$  – длина канала (толщина МКП). Поэтому среднее расстояние, которое проходит такой «средний» вторичный электрон до удара о стенку, равно ??

При выводе этого уравнения можно не учитывать составляющую начальной скорости электрона по оси канала Z.

Энергия, приобретенная «средним» электроном к моменту его столкновения со стенкой, составит ??

$$(10.1) ??$$

Можно считать, что на начальном участке кривой  $\delta(u_{\text{п}})$  КВЭ пропорционален энергии первичного электрона, а коэффициент пропорциональности зависит от угла падения. В нашей упрощенной модели усиления, которую можно назвать «шаговой», мы рассматриваем «средний» электрон со средним углом падения.

С другой стороны, среднее число стадий умножения равно отношению длины канала  $l_k$  к среднему расстоянию между соударениями  $l_e$ , т. е.

$$(10.2) ??$$

Поэтому коэффициент усиления канала  $G$  при «шаговой» модели усиления может быть определен как

$$(10.3) ??$$

Анализ этого уравнения показывает, что при росте напряжения на канале  $U_{\text{МКП}}$  повышается усиление  $G$ , так как повышается энергия  $\bar{u}_{\text{п}}$ , что приводит к росту  $\delta$ . В то же время увеличивается длина пролета электрона между ударами о стенку  $l_e$ , уменьшая число стадий

умножения  $\bar{m}_e$ . Поэтому в зависимости  $G(U)$  должен быть максимум. Однако при повышении усиления наступает насыщение, о котором будет сказано ниже, так что максимальное значение усиления не достигается.

В уравнении (10.3) имеется другой, более важный для понимания работы МКП максимум, который находится в зависимости  $G(\gamma_k)$ . С ростом калибра канала  $\gamma_k = l_k/d_k$  при постоянном напряжении на МКП  $U_{МКП}$  усиление в начале повышается из-за роста числа стадий умножения  $\bar{m}_e$ , а затем снижается из-за падения напряженности поля  $E$  и связанного с этим уменьшения КВЭ  $\delta$ .

Максимум усиления в зависимости  $G(\gamma_k)$  наступает при выполнении условия  $\frac{dG}{d\gamma_k} = 0$ .

После дифференцирования уравнения (10.3) и соответствующих преобразований получаем, что  $G = G_{\text{МАКС}}$  при  $\gamma_k = \gamma_{k,\text{ОПТ}}$ , причем

. Здесь  $e$  — основание натуральных логарифмов.

Величина  $A$  может быть определена из уравнений  $\delta = Au_{\text{П}}$ .

Учитывая, что  $u^*_0 = 1$  эВ, получим  $\gamma_{k,\text{ОПТ}} = \frac{U_{МКП}}{22}$ . Это означает, что максимальное усиление достигается в том случае, когда разность потенциалов между двумя точками по длине канала, отстоящими друг от друга на расстоянии его диаметра, равно постоянной величине 22 В.

В МКП длина всех каналов одинакова, однако их диаметр и соответственно калибр незначительно отличаются от канала к каналу из-за несовершенства технологии. В МКП с оптимальными параметрами структурный шум, определяемый различием усиления между отдельными каналами, минимален.

Полученная зависимость хорошо совпадает с экспериментальными данными.

Рассмотренная простейшая «шаговая» модель усиления в канале обладает рядом недостатков. Так, значения величин, рассчитанных по уравнениям (9.1), (9.2), (9.3), не полностью совпадают с экспериментальными данными.

*Модель Геста* учитывает статистику вторичноэлектронного умножения электронов, проявляющуюся в флуктуациях энергии вторичных электронов, углов их выхода с поверхности пластины, а

также количества соударений со стенками канала. Программа вычислений на ЭВМ, составленная с использованием методов Монте-Карло, предполагает последовательное определение каждого этапа процесса с учетом значений параметров умножения, выбранных методом генерации случайных чисел. Вычисления ведут для большого числа электронов, прослеживая все их пути и преобразования до выхода из канала. Вычисленные таким образом распределения параметров электронных сигналов на выходе из каналов, их средние и средние квадратичные значения детально описывают свойства МКП.

На рис. 10-2 показаны рассчитанные этим методом зависимости для энергии первичных электронов 2000 эВ. Кривые показывают, что отношение  $U_{МКП}/\gamma_k = 22$  В определяет максимальное усиление для всех значений  $\gamma_k$  и  $U_{МКП}$ .

Электрону, приходящему на вход канала, соответствует пучок вылетевших с выхода канала вторичных электронов, который принято называть *одноэлектронным импульсом*. Статистический характер процессов приводит к определенному статистическому распределению этих импульсов.

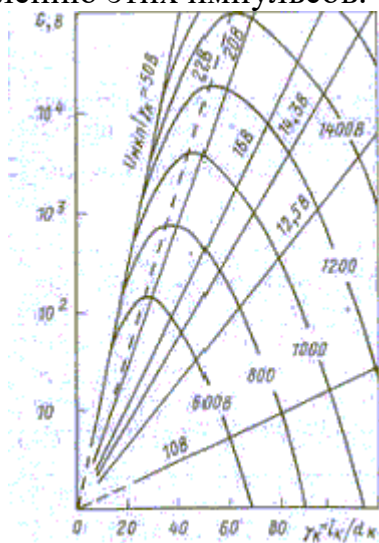


Рис. 9-2. Рассчитанная на ЭВМ зависимость электронного усиления МКП  $G$  от калибра канала  $\gamma_k=l_k/d_k$  для различных напряжений на МКП. Величина  $U_{МКП}/\gamma_k$  определяет падение напряжения на длине канала, равной его диаметру  $d_k$ .

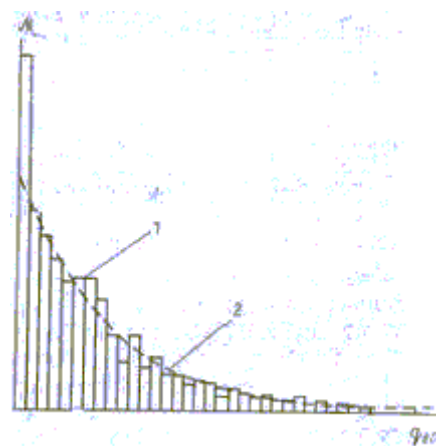


Рис. 9-3. Рассчитанная на ЭВМ (1) и экспериментально снятая (2) зависимость числа одноэлектронных импульсов  $N_{e0}$  от их амплитуды (заряда)  $q_{e0}$ .

На рис. 9-3 показана рассчитанная для компьютерной модели зависимость числа одноэлектронных импульсов МКП  $N_{e0}$  от их амплитуды (заряда)  $q_{e0}$  (кривая 1). Результаты расчета и эксперимента показывают, что это распределение соответствует отрицательной

экспоненте (кривая 2). Детальное изучение распределения свидетельствует, что в том случае, когда КВЭ для первичных электронов входного потока выше, чем КВЭ для электронов, умножаемых в канале, в области малых амплитуд появляется пик.

Время прохождения одноэлектронного импульса по каналу определяет среднее время задержки между двумя событиями — попаданием первичного электрона в канал и вылетом пучка электронов из него. Эту величину можно приближенно рассчитать, используя «шаговую» модель усиления. Очевидно, что среднее время прохождения одноэлектронного импульса по каналу  $T_k$  определяется произведением среднего времени пролета электрона до удара о стенку на среднее число стадий умножения: ??

Так, для МКП с диаметром каналов  $d_k = 40$  мкм при  $\gamma_k = 50$  и  $U_{МКП} = 1300$  В среднее время прохождения одноэлектронного импульса должно составить 0,5 нс.

Длительность одноэлектронного импульса может быть определена как время между двумя другими событиями — вылетом первых и вылетом последних электронов выходного пучка, т. е. как величина статистического разброса времени прохождения одноэлектронного импульса по каналу. Такую величину можно рассчитать, используя компьютерную модель.

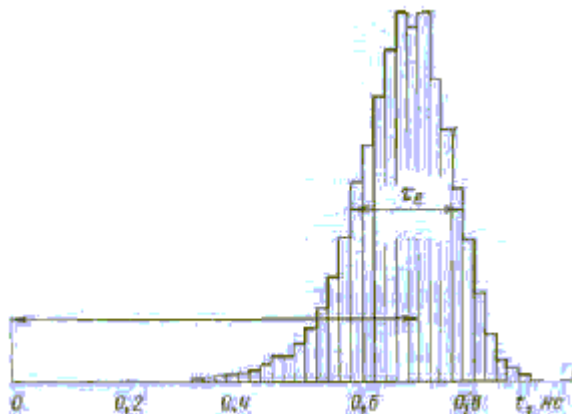


Рис. 9-4. Рассчитанное на ЭВМ распределение времени задержки в прохождении одноэлектронного импульса по МКП, форма и длительность одноэлектронного импульса:

$T_k$  — средняя величина времени задержки;  $t_e$  — средняя величина длительности одноэлектронного импульса

На рис. 9-4 показано вычисленное на ЭВМ распределение времени задержки в прохождении импульса по каналам МКП с  $d_k = 40$  мкм при  $\gamma_k = 50$  и  $U_{МКП} = 1300$  В. Среднее время задержки, т. е. среднее время прохождения одноэлектронного импульса  $T_k = 0,71$  нс, а

среднеквадратическое отклонение, т. е. средняя длительность одноэлектронного импульса  $\tau_e$  составляет 0,2 нс.

В пучке выходящих из канала вторичных электронов существенную долю составляют медленные электроны, порожденные на последней стадии умножения вблизи выхода из канала. Ряд вторичных электронов, возникших на ранних стадиях умножения и имеющих малую начальную энергию, не ударяет вновь по стенке канала, а разгоняется полем и покидает канал с большой скоростью. Средняя величина энергии в выходном пучке электронов должна быть близкой к энергии приобретенной «средним» электроном к моменту его соударения со стенкой. Величина средней энергии такого электрона составляет 70 эВ (при  $\gamma_k = 60$  и  $U_{МКП} = 1000$  В).

Распределение по энергиям электронов, выходящих из МКП, рассчитано для компьютерной модели с этими параметрами. Характер распределения близок к отрицательной экспоненте, при этом 25% электронов имеют энергию меньше 13,5 эВ, 50% – меньше 32,5 эВ, 75% – меньше 77 эВ. Незначительная доля электронов имеет при вылете энергию 200 ... 500 эВ.

Исследования показывают, что угловое распределение выходных электронов значительно более узко направлено, чем распределение Ламберта. Как первое, так и второе распределения существенным образом зависят от напряженности поля на выходе МКП, что свидетельствует о своеобразной фокусировке электронов на выходных участках каналов.

Следует специально отметить, что все приведенные выше данные для компьютерной модели МКП подтверждены экспериментальной проверкой с хорошим совпадением результатов.

Ряд примеров можно привести из области физики полупроводниковых приборов (например, модель идеального  $p-n$  перехода, выражающую зависимость тока через  $p-n$  переход от приложенного внешнего напряжения (см. *Литература* [13,14]).

## 10. Системы сбора данных

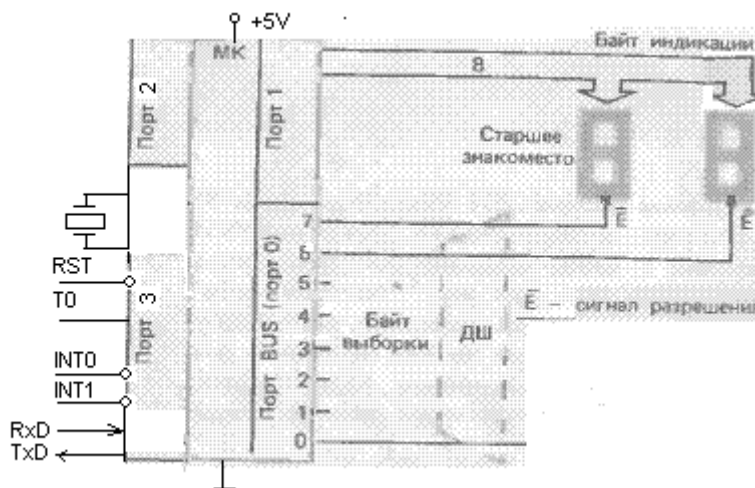
*Примеры. Простых и сложных систем сбора и обработки данных. Модель системы сбора данных и ее подсистемы. Две задачи. Пример 1 (проектирования системы с одним датчиком на базе микроконтроллера семейства MCS-51). Способы ввода данных. Пример 2 (система сбора данных на базе PC486). Интерфейсы PC. Пользовательский интерфейс.*

**Постановка задачи 1.** Необходимо разработать простую систему сбора данных на базе микроконтроллера. Система сбора данных включает в себя датчик температуры и цифровой индикатор.

*Упрощенный алгоритм работы контроллера.*

1. Инициализация системы
2. Считывание показаний датчика температуры (частота выходных импульсов пропорциональна температуре)
3. Вывод температуры на индикацию

В качестве микроконтроллера используется i8051 или какой-либо другой из семейства MCS-51. На INT0 подан сигнал внешнего генератора с частотой 1 с. На T0 подан частотный сигнал датчика.



**Постановка задачи 2.** Необходимо разработать систему сбора данных на базе PC486 (Буч, с.284). В системе должна быть предусмотрена возможность определения текущего времени и даты, которые будут использоваться при генерации сообщений о максимальных и минимальных значениях первичных параметров за последние 24 часа. Система должна обеспечивать постоянный вывод на дисплей текущих значений всех восьми первичных и производных параметров, а также текущее время и дату. Пользователь должен иметь возможность увидеть максимальные и минимальные значения любого из первичных параметров за 24 часа, сопровождаемые информацией о времени произведения соответствующего замера. Система должна позволять пользователю проводить калибровку датчиков по известным опорным значениям, а также устанавливать текущее время и дату.

Характеристика	Примечание
Система должна обеспечить автоматический съем	<u>Вычисляемые характеристики</u>
- температура	- относительное изменение температуры
- скорость и направление ветра	- коэффициент резкости погоды
- давление	- относительное изменение давления
- влажность воздуха	точка росы
Определение текущего времени, постоянный вывод на дисплей текущих значений всех параметров	Возможность увидеть на дисплее максимального и минимального значений первичных параметров за последние 24 часа.

## Анализ

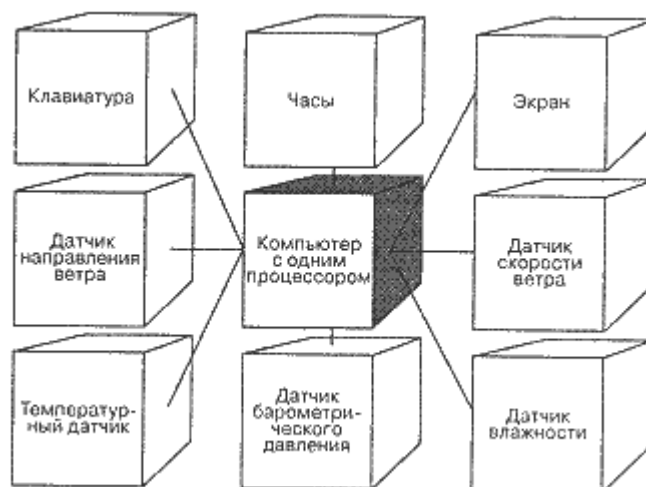
### Определение границ рассматриваемой задачи

На первый взгляд требования к системе мониторинга погоды довольно простая задача, решение которой позволяет обойтись всего несколькими классами. Инженер, не вполне искушенный во всех особенностях объектно-ориентированного анализа, может сделать поспешный вывод о том, что в данном случае наиболее простым и эффективным будет отказ от объектно-ориентированного подхода. Он обратится к рассмотрению потоков данных и входных/выходных значений. Тем не менее, как мы увидим в дальнейшем, даже для такой небольшой системы лучше ввести объектно-ориентированную архитектуру, которая прекрасно проиллюстрирует некоторые основные принципы, лежащие в основе объектно-ориентированной разработки.

Мы начнем наш анализ с рассмотрения аппаратной части системы. Эта задача системного анализа. Она включает в себя такие вопросы, технологичность и стоимость системы, которые выходят за рамки данной книги. Для того, чтобы сузить проблему, ограничившись анализом и проектированием только программных средств, сделаем следующие стратегические предположения об аппаратной части:

- Используется компьютер с одним процессором i486.
- Системные время и дата поддерживаются встроенными часами, соответствующие значения отображаются в оперативную память.
- Температура, барометрическое давление и влажность определяются встроенными контроллерами, которые соединены с соответствующими датчиками; показания контроллеров также отображены в оперативную память.
- Направление ветра измеряется с помощью флюгера с точностью до одного из 16 направлений; скорость ветра определяется анемометром со счетчиком оборотов.
- Ввод команд пользователем осуществляется с помощью клавишной панели (похожей на телефонную), сигналы которой обрабатываются с помощью встроенной платы. Эта же плата обеспечивает звуковой сигнал после каждого нажатия клавиши. Последняя команда пользователя сохраняется в памяти.
- Экраном служит обычный жидкокристаллический дисплей. Встроенный контроллер дисплея обеспечивает вывод на экран небольшого набора графических примитивов: линий и дуг, закрашенных областей и текста.
- Встроенный таймер посылает прерывания через каждую 1/60 долю секунды.

На рис. 1 приведена диаграмма, иллюстрирующая состав аппаратной части системы.





## **Проектирование**

### ***Архитектурный каркас***

Каждая программная система должна иметь простую и в то же время всеобъемлющую организационную философию. Система мониторинга погоды не является в этом смысле исключением. На следующем этапе нашей работы мы должны четко определить архитектуру проекта. Это даст нам стабильный фундамент, на основе которого мы будем строить отдельные функциональные части системы.

Существует целый ряд архитектурных моделей для решения задач сбора и обработки данных и управления, но наиболее часто встречаются синхронизация автономных исполнителей и схема покадровой обработки.

В первом случае архитектура системы сконструирована из ряда относительно независимых объектов, каждый из которых выполняется как поток управления. Можно было бы, например, создать несколько новых объектов-датчиков, построенных с помощью более примитивных абстракций, каждый из которых отвечал бы за считывание информации с определенного датчика и за передачу ее центральному агенту, обрабатывающему всю информацию. Подобная архитектура имеет свои преимущества и является, пожалуй, единственной приемлемой моделью в случае проектирования распределенной системы, которая должна производить обработку большого числа параметров, поступающих с удаленных датчиков. Эта модель также позволяет эффективнее оптимизировать процесс сбора данных (каждый объект-датчик может содержать в себе информацию о том, как надо приспособляться к изменению окружающих условий — увеличивать или уменьшать частоту опроса, например).

### ***Эволюция***

#### ***Планирование релизов***

Рассмотрев несколько сценариев работы системы и убедившись в правильности стратегических решений, можно начинать планирование процесса разработки. Разобьем работу на ряд этапов, результат каждого из которых будет являться основой для последующего:

- Разработка программы, обладающей минимальными функциональными свойствами и осуществляющей мониторинг только одного датчика.
- Создание иерархии датчиков.
- Создание классов, ответственных за управление изображением на экране.
- Создание классов, ответственных за работу пользовательского интерфейса.

В принципе, можно было бы изменить порядок этапов, но мы выбрали именно такую последовательность, исходя из того, что наиболее сложная и рискованная часть работы должна выполняться в первую очередь.

Разработка минимальной версии программы заставляет нас в первую очередь смоделировать архитектуру «по вертикали», реализовав в усеченном варианте практически все ключевые абстракции. Эта задача несет в себе основной риск, ведь в процессе ее решения фактически проверяется правильность выбора ключевых абстракций, их роль и функции. Успешное создание раннего прототипа играет очень большую роль в построении системы. Это дает нам ряд технических (и не только) преимуществ. В частности, мы сразу выявим несоответствия между аппаратной и программной частями. Кроме того, будущие пользователи получат возможность уже на ранних этапах проекта оценить внешний вид и работу системы.

#### ***Литература***

1. Буч Г.. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. Изд.2 М.: Издательство Бином , 1999. - 560 с.
2. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах. М.: Энергоиздат .1990.
3. Гук М. Интерфейсы ПК. Питер. 1999.

## 11. Системы массового обслуживания

*Историческая справка.* Толчком к возникновению теории массового обслуживания (ТМО) стали работы датского инженера и математика А.К. Эрланга, опубликованные в 1908–1922 гг. Первоначально ТМО была направлена на решение задач, касающихся вопросов обслуживания абонентов телефонной станции, расчета запасов магазинов для бесперебойного снабжения покупателей, а также установления наиболее рационального числа продавцов и касс в торговых предприятиях.

В условиях современного информационного бума хозяином жизни является тот, кто имеет доступ к информации различного рода. Такое положение естественно приводит к необходимости разработки различных систем вероятностной природы, передающих и обрабатывающих информацию. В качестве примеров можно назвать сети связи, разнообразные вычислительные системы, системы передачи данных и др.

Разработка и проектирование таких систем, в свою очередь, требуют математического анализа некоторых обобщенных вероятностных моделей, достаточно адекватно описывающих процесс их функционирования. Действительно, в общем случае каждую из подобных систем можно интерпретировать как некоторый механизм обработки требований, сообщений или заявок, образующих случайный поток. Например, каждый узел сети связи представляет собой механизм обработки сообщений, поступающих в него в некоторые случайные моменты времени; компьютерная сеть представляет собой механизм обработки случайного потока программ пользователей и т.д.

Теория массового обслуживания (ТМО), или, иначе, теория очередей, представляет собой раздел теории вероятностей, который занимается исследованием процессов обслуживания, т.е. специальным классом случайных процессов.

В ТМО анализируются характеристики информационных систем (систем обработки информации), в которые информация поступает порциями. Эти порции в ТМО принято называть требованиями. Требования обрабатываются различными устройствами; время обработки (обслуживания) требования, вообще говоря, представляет собой случайную величину (СВ). Подобные системы, составляющие предмет ТМО, называются системами массового обслуживания (СМО).

СМО представляют собой модели большого количества реальных систем, например, коммутационных систем, компьютерных сетей, очередей в магазинах, транспортных перекрестков и т.д. В состав СМО входят следующие элементы.

**1. Входной поток требований.** Этот элемент является одним из основных, его обязательно содержит каждая СМО. Входным потоком называется последовательность моментов времени, в которых требования с целью дальнейшего обслуживания (т.е. некоторой обработки) поступают на вход СМО. Указанные моменты, как правило, являются случайными, хотя иногда могут быть и фиксированными (в этом случае поток называется детерминированным или регулярным).

Если поступление требования в систему трактовать как случайное событие, то входной поток является потоком случайных событий или, короче, случайным потоком. Примерами случайных потоков являются последовательность программ пользователя, поступающих на вход компьютера, поток вызовов,

поступающих на телефонную станцию, или поток команд, поступающих в устройство управления компьютера.

**2. Механизм обслуживания.** Задание механизма обслуживания подразумевает определение того, в каких условиях обслуживание требования возможно, сколько требований может обслуживаться одновременно и как долго длится обслуживание требования. Длительность времени обслуживания в общем случае представляет собой случайную величину и характеризуется определенным статистическим распределением. Ресурс (устройство), обслуживающий требования, называется обслуживающим прибором (ОП).

**3. Дисциплина обслуживания.** Под дисциплиной обслуживания понимают правило, в соответствии с которым устанавливается очередность обслуживания поступающих в систему требований. Например, требования могут обслуживаться в порядке их поступления в систему, либо в обратном (инверсионном) порядке, когда при освобождении ОП на обслуживание принимается требование, которое поступило в систему последним, либо в соответствии с некоторым фиксированным или динамическим приоритетом и т.д.

**4. Очередь.** В некоторых СМО очередь возникает в том случае, если в момент поступления требования начало его обслуживания невозможно. В этом случае требование может потеряться или ожидать обслуживания в очереди (в зависимости от конкретной анализируемой модели). Очередь характеризуется числом мест ожидания, т.е. максимальным числом требований, которые могут находиться в этой очереди одновременно. Если число мест ожидания в очередях СМО равно нулю, то это система с потерями. Если каждое поступившее требование обслужено полностью, то это СМО без потерь.

**5. Выходной поток требований** представляет собой последовательность моментов времени окончания обслуживания требований (моментов времени окончания обслуживания требований или, иначе, моментов выхода требований из системы). Анализ такой последовательности важен, если выходной поток является входным по отношению к другой СМО.

**Теорема Литтла.** Рассмотрим систему массового обслуживания, в которой требования на обслуживание появляются в случайные моменты времени. Среднее число требований в системе  $N$  и средняя задержка  $T$  связаны простой формулой, которая дает возможность выражать одно среднее через другое. Этот результат, известный как теорема Литтла, имеет вид  $N = \lambda T$ .

### **Задержки информационных пакетов [3]**

**Время распространения** — промежуток времени от момента, когда последний бит был передан на начальном узле линии, до момента, когда он будет принят в конечном узле этой линии. Эта задержка пропорциональна физическому расстоянию между передатчиком и приемником и обычно мала, за исключением случая спутниковой линии связи. При таком рассмотрении не учитывается возможность того, что может потребоваться повторная передача по линии из-за ошибок передачи или каких-либо других причин. Для большинства реальных линий связи, за исключением линий множественного доступа, повторные передачи бывают редко и они не будут рассматриваться. Время распространения зависит от физических характеристик линии связи и не зависит от потока, передаваемого по линии. Задержка обработки также не зависит от величины потока, обрабатываемого в соответствующем узле, если вычислительная мощность узла не является ограниченной. Мы примем это

предположение в наших рассуждениях. В противном случае отдельная очередь на обработку должна быть введена перед очередями на передачу. Большая часть нашей последующего анализа сосредоточена на задержках в очереди и задержках передачи. Сначала мы рассмотрим единственную линию связи и проанализируем некоторые классические модели теории массового обслуживания. Затем рассмотрим сеть и обсудим те приближения, которые допускаются, когда выбираются аналитические модели для расчета задержки.

Хотя главный акцент делается на модели сети с коммутации пакетов, некоторые разработанные модели пригодны также для сети с коммутацией каналов. Теория массового обслуживания широко развивалась исходя из потребностей создания математических моделей для телефонии.

## **12. Система охранной сигнализации**

*Постановка задачи. Примеры реализации и применения*

В общем случае необходимо разработать систему, реагирующую на попытки проникновения на охраняемую территорию и/или похищения охраняемого объекта.

**Пример 1.** Система на базе персонального компьютера (ноутбука) с использованием клавиатуры в качестве устройства сопряжения с датчиками. С точки зрения проектирования такой системы можно использовать опыт моделирования системы сбора данных, рассмотренной в лекции 10.

**Пример 2.** Система охранной сигнализации (СОС) для установки на автомобиле на базе однокристального микроконтроллера. СОС должна немедленно оповещать при открывании капота, крышки багажника или любой из трех дверей, за исключением двери водителя. При открывании двери водителя СОС делает 10-секундную выдержку и, если за это время СОС не будет отключена путем набора секретной комбинации цифр (например, 74) и нажатия кнопки ввода ВВ, СОС должно включать тревожную сигнализацию. Тревожная сигнализация включается и в том случае, если набран неправильный код (отличный от секретного числа). Включение или отключение СОС производится потайным выключателем электропитания микроконтроллера.

**Анализ задачи и разработка аппаратной части.** Выбор микроконтроллера (i8048, i8051, AT89S53)

**Разработка алгоритма.** (См рис.)

Инициализация МК

Цикл ожидания начала работы

Режим охраны

Подпрограмма задержки на 10 с.

**Запись программы в память МК и ее отладка.**

**Испытания системы и возможность ее развития.**

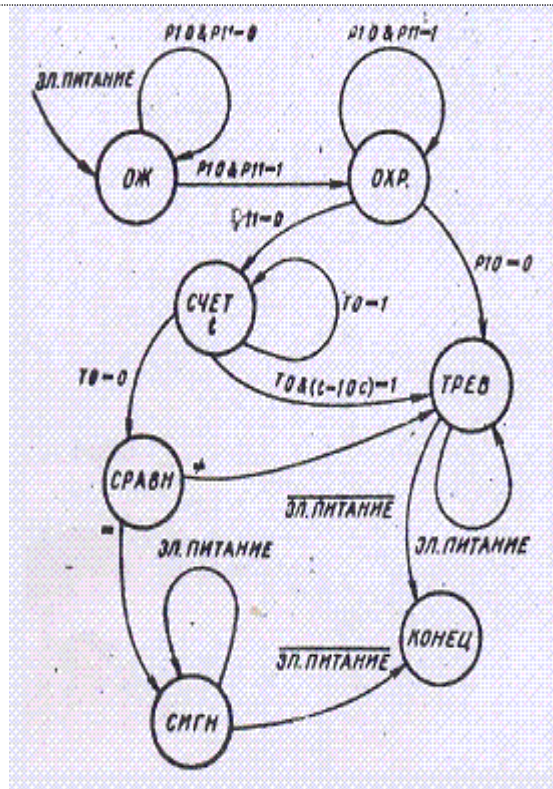


Рис. Граф автомата СОС

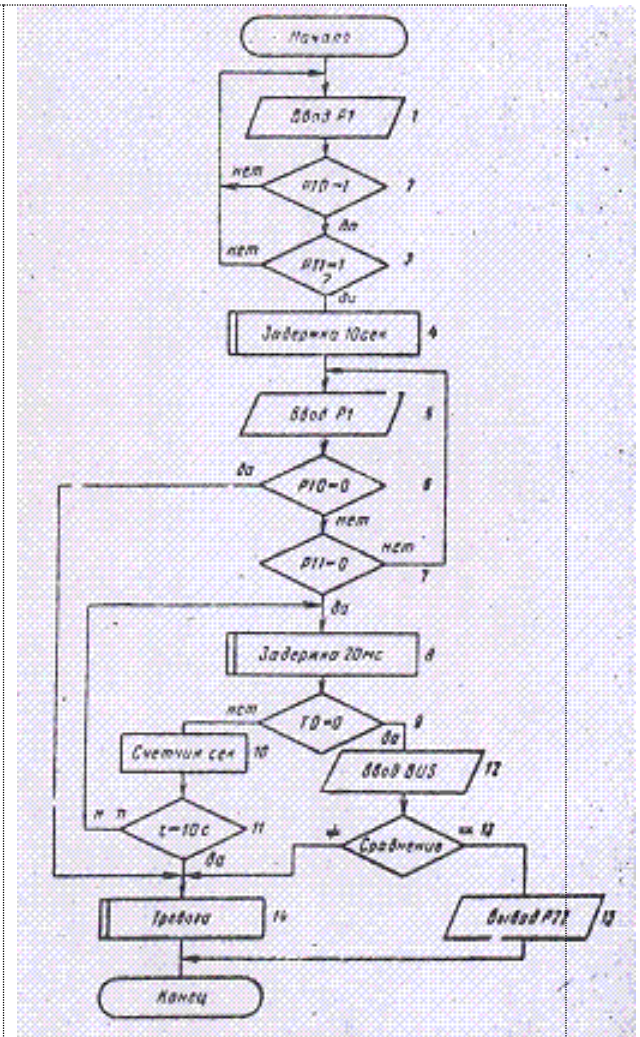


Рис. Блок-схема алгоритма работы

*“Учение без размышления бесполезно, но  
и размышление без учения опасно”*  
Конфуций

*“Нам без отдыха и срока жужжат в уши,  
сообщая разнообразные знания, в нас вливают  
их, словно воду в воронку, и наша обязанность  
состоит лишь в повторении того, что мы  
слышали. Я хотел бы, чтобы воспитатель  
вашего сына отказался от этого приема ... ”*  
М. Монтень

### **Заключение**

В этом специальном курсе основное внимание было направлено на методологию (объектно-ориентированный анализ и моделирование) наиболее распространенные практические методы моделирования, а также примеры из предметной области.

Разработка сложных программных моделей выполняется большими коллективами. В таких коллективах работают не только программисты, но также и те, кто наряду с программированием должен обладать необходимыми знаниями из предметной области. Те, кто понимает, чего хотят специалисты и как поставить задачу программисту. Например, для создания программной модели лавинного фотодиода или микроканальной пластины недостаточно быть программистом (знать Matlab, Pascal или C++). Здесь не обойтись без понимания тонкостей устройства и работы этих приборов. Рассмотренная методология создает необходимые условия для успешного взаимодействия программистов и специалистов из различных предметных областей.

Следует отметить, что моделирование - достаточно универсальный инструмент, поэтому методы, изученные в этом спецкурсе, могут быть использованы не только в области радиофизики и электроники.

В заключение курса отметим наиболее важные моменты. Построение модели проектируемой системы (электронной или программной) – необходимая предпосылка, обеспечивающая успех всего проекта.

## Литература

### *а) основная*

1. Буч Г.. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. Изд.2 М.: Издательство Бином , 1999. - 560 с.
2. Буч Г., Рамбо Д., Джекобсон А. UML .Универсальный язык моделирования. Руководство пользователя.
3. Бертсекас Д., Галлагер Р. Сети передачи данных. М. Мир 1989. 544 с.
4. Гулд Х., Тобочник Я. Компьютерное моделирование в физике: В 2-х частях. Пер с англ. М.: Мир, 1990.
5. Allen M, Tildesley D. Computer simulation of liquids. 1991.
6. Frenkel, D.; Smith, V. Understanding Molecular Simulations: from Algorithms to Applications. Academic Press; San Diego, 1996.
7. Бенькович Е. и др. Практическое моделирование динамических систем. СПб. БХВ. 2002.
8. Гроп Д. Идентификация систем. М.: Мир.
9. Кузнецов С.П. Динамический хаос. М.: Физматлит, 2001. – 296 с.

### *б) дополнительная*

10. Неймарк Ю. О простых моделях сложных систем (статья).
11. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристалльных микроконтроллерах. М.: Энергоиздат .1990.
12. Гук М. Интерфейсы ПК. Питер. 1999.
- [11]. Апанасович В.В., Тихоненко О.М. Цифровое моделирование стохастических систем. Мн.: Университетское. 1986 – 127 с.
- [12]. Мулярчик С.Г. Численное моделирование микрoэлектронных структур. - Минск: Изд-во Университетское. 1987. - 368 с.
- [13]. Зи С. Физика полупроводниковых приборов. В 2-х книгах. М. Мир. 1984.
- [14]. Сугано Т, Икома Т., Такэиси Е. Введение в микроэлектронику: Пер. с яп. - Мир, 1988.
15. Либерти Дж. Освой самостоятельно C++ за 21 день: 3-е изд. М.: 2000. - 815 с.
16. Советов Б.Я., Яковлев С.А. Моделирование систем. М.: Высшая школа, 1985. 271 с.

### *в) электронные документы*

(???? см. файл 2004 umk MPS 15v1 лекция 10)

## Приложение 1.

### Лабораторная работа № 1

#### Построение моделей линейных динамических систем

**Цель работы:** моделирование динамических систем с непрерывным временем

**Задачи:** 1) Изучение форм представления моделей линейных динамических систем; 2) определение зависимости состояния системы от времени при заданном векторе возмущающих воздействий; 3) повторение и развитие навыков практической работы с пакетом MATLAB/Signal Processing Toolbox.

#### Основные сведения.

В данной работе рассматриваются модели линейных динамических систем (механических, электрических и др.), которые могут быть описаны системой дифференциальных уравнений.

Модели линейных систем можно представить в следующих формах:

- State-space - пространство состояний
- Transfer function - передаточная функция
- Zero-pole - нулей-полюсов (нули – корни алгебраического выражения в числителе, полюса – корни выражения в знаменателе передаточной функции).

В пространстве состояний модель представляют в виде:

$$\dot{x} = Ax + Bu$$

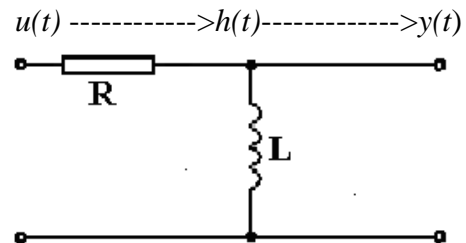
$$y = Cx$$

$x$  - вектор состояния;  $u$  - вектор возмущающих воздействий;  $A$  - матрица определяемая системой дифференциальных уравнений;  $B$  - матрица определяющая управляемость системы;  $C$  - матрица определяющая наблюдаемость системы.

Решать данную систему дифференциальных уравнений можно с помощью преобразования Лапласа, осуществляющему переход к алгебраическим уравнениям вида:

$$Y(s) = H(s)u(s) \quad (*)$$

**Пример** моделирования простейшей системы:



(Конкретный вид системы указан в индивидуальном задании)

Передаточная функция такой системы имеет вид:

$$H(s) = \frac{sL}{sL + R}.$$

Программа, представленная ниже, осуществляет переход в пространство состояний (функция tf2ss), а также в пространство нулей-полюсов передаточной функции (функция tf2zp).

Аналогично можно перейти к представлению передаточной функции в виде нулей и полюсов.

Способ описания системы в форме передаточной функции эквивалентен способу описания системы в пространстве состояний, причем в радиоэлектронике он используется чаще.

**Порядок выполнения:** 1. Определить передаточную функцию и преобразование Лапласа от входного воздействия  $u(t)$  (по умолчанию входное воздействие – импульс единичной амплитуды и единичной длительности) 2) С помощью соотношения (\*) определить сигнал на выходе системы как функцию переменной  $s$ , а затем – как функцию времени (с помощью обратного преобразования Лапласа). 3) Построить график  $y(t)$  для  $0 < t < 2$ .



## Лабораторная работа № 2. Метод молекулярной динамики

```
clear all
% script moldy2 for N = 2 mol dynamics ( nstep=100) % function used: forc2, sepa2, vex, vey
N=2;
mas=[1 1];
Lx=8;
Ly=8;
vmax=0.3;
dt=0.01;
dt2=dt*dt;
nstep=100;
x0=[3.5,4.5];
y0=[3.5,4.5];
v0=[vmax*0.707,-vmax*0.707];
% vy0=[vmax*0.707,-vmax*0.707];
xt=x0;
yt=y0;
dx0=xt(2)-xt(1);
dy0=yt(2)-yt(1);
dxy0=dx0.*dx0+dy0.*dy0;
r0=sqrt(dxy0);
xx=0:Lx;
yy=0:Ly;
figure(1)
plot(xx,yy)
hold on
plot(x0,y0,'r*')
title('* - molecules start position')
hold off
f0=forc1(r0);
ac=f0./mas;
ac(2)=-ac(1);
x=x0;
y=y0;
vt=v0;
for k=1:nstep
xt=vex(x,vt,ac,dt);
yt=vey(y,vt,ac,dt);
figure(2)
plot(xx,yy)
hold on
plot(x0,y0,'r*')
plot(xt,yt,'g+')
title('molecules current+ and start* positions')
hold off
step=k
v=vt
acold=ac;
x=xt;
y=yt;
r=sepa2(x,y)
f2=forc2(r);
ac=f2./mas;
ac(2)=-ac(1);
vt=v+(acold+ac). *dt.*dt/2;
vt(2)=-vt(1);
end
```

### Приложение 3. Лабораторная работа № 3

#### Моделирование электронного прибора

**Цель работы:** разработка модели электронного умножителя

**Задачи:** 1) моделирование одноэлектронного импульса 2) получение распределения (гистограммы) выходных импульсов. 3) теоретическая оценка и сравнение (теория--эксперимент) по критерию хи-квадрат.

**Исходные данные.** Электронный умножитель состоит из  $M$  каскадов. Число вторичных электронов в каждом каскаде носит вероятностный характер и подчиняется распределению Пуассона с параметром  $m$  (при воздействии на вход умножителя одиночного электрона). Каждая из полученных частиц ударяет следующий диод, и независимо друг от друга электроны выбивают какое-то количество частиц, которое подчиняется распределению Пуассона.

Таким образом, требуется реализовать следующий алгоритм:

1. Задать параметры  $M$  и  $m$

2. Моделирование умножителя реализуется в цикле. Первоначально на входе один электрон. С помощью функции Пуассона `poissrnd(m)`, моделируется количество электронов на выходе после первого диода. Затем на второй итерации повторить вызов этой функции для всех частиц (количество вызовов равно выходу первого каскада). Суммировать количество электронов на выходе этого каскада. Так продолжается для каждого диода.

3. Если каскад последний, суммарное количество электронов заносится в массив выходных данных

4. Визуализация результатов моделирования.

#### Выполнение работы: индивидуальные задания

Вариант\ параметры	$M$	$m$	$N$	Примечание
1	3	3	500	
2	4	2	200	

Листинг 1

#### РЕАЛИЗАЦИЯ ПРОГРАММЫ НА MATLAB

```
clear all;
clc;
% functions used for generates Poisson random digits:  poissrnd(m)
m = 3;
NumOfCascades = 3;
N = 100;
.....
```

### Приложение 4. Лабораторная работа № 4

Имитационное моделирование дискретных процессов в фотофизике

**Цель работы:** моделирование дискретного фотофизического процесса методом Монте Карло

**Задачи:** 1) Изучение физического базиса флуоресценции; 2) выработка формального подхода к моделированию; 3) реализация модели; 4) верификация модели

**Постановка задачи и теоретический базис.** Даны в электронной версии лекции.

**Варианты заданий.**

1. (сложность \*\*) Смоделировать флуоресценцию смеси 2х флуорофоров во временной и спектральной области. Параметры системы:

Параметр	вариант А	вариант Б	вариант В	вариант Г
$\tau_1$	3 нс	1 нс		
$\tau_2$	3 нс	5 нс		
$q_1$	0.9	0.8		
$q_2$	0.5	0.7		
$\varepsilon_1$	800 М <sup>-1</sup> см	1000 М <sup>-1</sup> см		
$\varepsilon_2$	1200 М <sup>-1</sup> см	1200 М <sup>-1</sup> см		
$\lambda_1$	300 нм	350 нм		
$\lambda_2$	400 нм	600 нм		
$\sigma_1$	2500 см <sup>-1</sup>	2000 см <sup>-1</sup>		
$\sigma_2$	3000 см <sup>-1</sup>	3000 см <sup>-1</sup>		
$C_1$	1 мкМ	1 мкМ		
$C_2$	1.5 мкМ	1.2 мкМ		

При моделировании считать, что молекулы обладают гауссовым уширением спектра испускания с положением максимума флуоресценции  $\lambda_i$  и шириной линии  $\sigma_i$ . При моделировании спектра следует пользоваться характеристиками пропорциональными энергии (частотой), а конечный результат представлять в виде зависимости  $s(\lambda)$ .

## Приложение 5. Лабораторная работа №5

### Моделирование системы массового обслуживания (СМО)

**Цель работы:** построение модели СМО и оценка точности результатов моделирования

**Задачи:** 1) моделирование входного потока, времени обслуживания и выходного потока (см. инд. задание) – **1 балл**;

2) определение заданных характеристик системы (коэффициент простоя канала) и выходного потока (интенсивность распределение) – **3 балла**;

3) сравнение результатов компьютерного моделирования с аналитическим выражением (\*).

Основные сведения. В данной работе рассматриваются модели СМО типа M/D/1, M/M/1, G/D/n, G/M/n. Коэффициент простоя определяется как отношение суммарного время простоя системы (канала) к полному времени моделирования. Теоретическое значение коэффициента простоя (в случае экспоненциального распределения интервалов между событиями входного потока и времени обслуживания) равно  $Kпр = 1 - \lambda / \nu$ , \*) где  $\lambda$  и  $\nu$  – интенсивности входного потока и обслуживания соответственно, причем интенсивность обслуживания  $\nu$  – величина обратная среднему времени обслуживания одного требования.

Привести текст программы, результат её прогона, оценки заданных характеристик, сравнение результатов моделирования с теорией (критерий хи-квадрат), в комментарии указать дату выполнения.

**Теория СМО.** Разработка и проектирование систем требуют математического анализа некоторых обобщенных вероятностных моделей, достаточно адекватно

описывающих процесс их функционирования. Действительно, в общем случае каждую из подобных систем можно интерпретировать как некоторый механизм обработки требований, сообщений или заявок, образующих случайный поток. Например, каждый узел сети связи представляет собой механизм обработки сообщений, поступающих в него в некоторые случайные моменты времени; компьютерная сеть представляет собой механизм обработки случайного потока программ пользователей и т.д.

Теория массового обслуживания (ТМО), или, иначе, теория очередей, представляет собой раздел теории вероятностей, который занимается исследованием процессов обслуживания, т.е. специальным классом случайных процессов.

В ТМО анализируются характеристики информационных систем (систем обработки информации), в которые информация поступает порциями. Эти порции в ТМО принято называть требованиями. Требования обрабатываются различными устройствами; время обработки (обслуживания) требования, вообще говоря, представляет собой случайную величину (СВ). Подобные системы, составляющие предмет ТМО, называются системами массового обслуживания (СМО).

СМО представляют собой модели большого количества реальных систем, например, коммутационных систем, компьютерных сетей, очередей в магазинах, транспортных перекрестков и т.д. В состав СМО входят следующие элементы.

**1. Входной поток требований.** Этот элемент является одним из основных, его обязательно содержит каждая СМО. Входным потоком называется последовательность моментов времени, в которых требования с целью дальнейшего обслуживания (т.е. некоторой обработки) поступают на вход СМО. Указанные моменты, как правило, являются случайными, хотя иногда могут быть и фиксированными (в этом случае поток называется детерминированным или регулярным).

Если поступление требования в систему трактовать как случайное событие, то входной поток является потоком случайных событий или, короче, случайным потоком. Примерами случайных потоков являются последовательность программ пользователя, поступающих на вход компьютера, поток вызовов, поступающих на телефонную станцию, или поток команд, поступающих в устройство управления компьютера.

**2. Механизм обслуживания.** Задание механизма обслуживания подразумевает определение того, в каких условиях обслуживание требования возможно, сколько требований может обслуживаться одновременно и как долго длится обслуживание требования. Длительность времени обслуживания в общем случае представляет собой случайную величину и характеризуется определенным статистическим распределением. Ресурс (устройство), обслуживающий требования, называется обслуживающим прибором (ОП).

**3. Дисциплина обслуживания.** Под дисциплиной обслуживания понимают правило, в соответствии с которым устанавливается очередность обслуживания поступающих в систему требований. Например, требования могут обслуживаться в порядке их поступления в систему, либо в обратном (инверсионном) порядке, когда при освобождении ОП на обслуживание принимается требование, которое поступило в систему последним, либо в соответствии с некоторым фиксированным или динамическим приоритетом и т.д.

**4. Очередь.** В некоторых СМО очередь возникает в том случае, если в момент поступления требования начало его обслуживания невозможно. В этом случае требование может потеряться или ожидать обслуживания в очереди (в зависимости от конкретной анализируемой модели). Очередь характеризуется числом мест ожидания, т.е. максимальным числом требований, которые могут находиться в этой очереди одновременно. Если число мест ожидания в очередях СМО равна нулю, то это система с потерями. Если каждое поступившее требование обслужено полностью, то это СМО без потерь.

**5. Выходной поток требований** представляет собой последовательность моментов времени окончания обслуживания требований (моментов времени окончания обслуживания требований или, иначе, моментов выхода требований из системы). Анализ такой последовательности важен, если выходной поток является входным по отношению к другой СМО.

## Приложение 6. Код программы MD2 (Pascal) – к лаб. работе № 2

```
program md2 (input, output);
{$N+}
const   c = 32768;                               (* for random geration *)
        Nmax =100;                               (* Maximum of particles *)
type
  component = array[1..Nmax] of real;
var
  x, y, vx, vy, ax, ay      : component;
  xCentr, yCentr            : real;
  N, nave, nset, iset, iave : integer;
  Lx, Ly, dt, dt2          : real;
  virial, xflux, yflux, pe, ke, time : real;

{=====}
procedure initial (var x, y, vx, vy : component;
var N, nave, nset : integer; var Lx, Ly, dt, dt2 : real);
var irow, icol, nrow, ncol, i : integer;
    ax, ay, xscale, yscale, Mx, My : real;
    vscale, vmax, vxcum, vycum : real;
    fname : string;
    datain : text;
    newconf : char;
begin
  write('number of molecules = ');
  readln(N);
  writeln('Box dimention');
  write('Lx = ');
  readln(Lx);
  write('Ly = ');
  readln(Ly);
  write('Shag po vremeni = ');
  readln(dt);
  dt2 := dt * dt;
  write('Shagov mezdu avereg, nave = ');
  readln(nave);
  write('Naborov usredneniya, nset = ');
  readln(nset);
  write('New config? (y/n)');
  readln(newconf);
  if (newconf = 'y') then
    begin
      (* na treugolnoj reshetke *)
      vxcum := 0;
      vycum := 0;
      write('n molecul in row, nrow = ');
      readln(nrow);
      write('max skorost', vmax = ');
      readln(vmax);
      ncol := N div nrow;
      (* rasstoyanie do bliжайshego soseda v kajdom napravlenii *)
      ay := Ly / nrow;
      ax := Lx / ncol;
      i := 0;
      for icol := 1 to ncol do
        for irow := 1 to nrow do
          begin
            i := i + 1;
            y[i] := ay * (irow - 0.5);
```

```

    if ((irow mod 2) = 0) then
        x[i] := ax * (icol - 0.25)
    else
        x[i] := ax * (icol - 0.75);
        vx[i] := (2*random - 1) * vmax;      (* v -random *)
        vy[i] := (2*random - 1) * vmax;
        (* sum Vx and Vy = 0 ? *)
        vxcum := vxcum + vx[i];
        vycum := vycum + vy[i];
        write('x',i,'=',x[i]:6:3);
        writeln(' y',i,'=',y[i]:6:3);
    end;

readln;

vxcum := vxcum / N;
vycum := vycum / N;

for i := 1 to N do begin
    vx[i] := vx[i] - vxcum;
    vy[i] := vy[i] - vycum
end;
end

else      (* old configur *)
begin
write('filename OLD configur, fname');
readln(fname);

{ write('otnositelnoe izmenenie skorost, vscale = '); }
{ readln(vscale); }
vscale := 1.0;

assign(datain, fname);
reset(datain);
readln(datain, N, Mx, My);
xscale := Lx / Mx;
yscale := Ly / My;
for i := 1 to N do begin

    readln(datain, x[i], y[i], vx[i], vy[i]);
    x[i] := x[i] * xscale;
    y[i] :=y[i]* yscale;
    vx[i] := vx[i] * vscale;
    vy[i] := vy[i] * vscale end;
close( datain)
end
end;

{=====
=====}
procedure periodic (var xtemp, ytemp, xflux, yflux : real; px,
py, Lx, LY : real);
begin
if xtemp < 0.0 then begin
xtemp:= xtemp + Lx;
xflux := xflux - px

```

```

end;

if xtemp > Lx then begin
    xtemp := xtemp - Lx;
    xflux := xflux + px
end;

if ytemp < 0.0 then begin
    ytemp := ytemp + Ly;
    yflux := yflux - py
end;

if ytemp > Ly then begin
    ytemp := ytemp - Ly;
    yflux := yflux + py
end
end;
{=====
=====}
procedure separation (var dx, dy : real; Lx, Ly : real);
begin
if abs(dx) > 0.5 * Lx then
    dx := dx * (1.0 - Lx / abs(dx));
if abs(dy) > 0.5 * Ly then
    dy := dy * (1.0 - Ly / abs(dy)) end;

{=====
=====}
procedure f (r : real; var force, potential : real);
var ri, ri3, ri6, g : extended;
begin
ri := 1.0 / r;
ri3 := ri * ri * ri;
ri6 := ri3 * ri3;
g := 24.0 * ri * ri6 * (2.0 * ri6 - 1.0);
{writeln('g=',g);}
force := g * ri;
potential := 4.0 * ri6 * (ri6 - 1.0)
end;
{=====
=====}
procedure accel (var x, y, ax, ay : component; N : integer; Lx,
Ly : real; var pe : real);
var i, j : integer;
dx, dy, r, force, potential : real;
begin
for i := 1 to N do begin
    ax[i] := 0.0;
    ay[i] := 0.0
end;

for i := 1 to (N - 1) do
    for j := (i + 1) to N do begin
        dx := x[i] - x[j];
        dy := y[i] - y[j];
        separation(dx, dy, Lx, Ly);
        r := sqrt(dx * dx + dy * dy);
        f(r, force, potential);
    
```



```

        ax[i] := ax[i] + force * dx;
        ay[i] := ay[i] + force * dy;
        ax[j] := ax[j] - force * dx;
        ay[j] := ay[j] - force * dy;
        pe := pe + potential
    end
end;
{=====
=====}
procedure Verlet (var x, y, vx, vy, ax, ay : component; N :
integer; Lx, Ly, dt, dt2 : real;
var virial, xflux, yflux, pe, ke : real);
var
i : integer; xnew, ynew : real;
begin
for i := 1 to N do begin
    xnew := x[i] + vx[i] * dt + 0.5 * ax[i] * dt2;
    ynew := y[i] + vy[i] * dt + 0.5 * ay[i] * dt2; (* sacTHMHO MSHseM
CKOpocrb, ncnoJib3yH crapoe ycKopeHHe *)
    vx[i] := vx[i] + 0.5 * ax[i] * dt;
    vy[i] := vy[i] + 0.5 * ay[i] * dt;
    (* nepnoflMMecKHe xpaeebie ycnOBMs h BbmwciiEHne noTOKa *)
    periodic( xnew, ynew, xflux, yflux, vx[i], vy[i], Lx,
Ly);
    x[i] := xnew; y[i] := ynew;
    { write('x',i,'=',x[i]:6:3);
      write(' y',i,'=',y[i]:6:3);
      readln; }
end;
accel(x, y, ax, ay, N, Lx, Ly, pe); (* pac4eT HOBoro ycKopenia
*)
{writeln('ke = ',ke);}
for i := 1 to N do (* OKOH^aTemHO MewieM CKopocrt, wcnoribcya hobos
ycKopeHHe *) begin
    vx[i] := vx[i] + 0.5 * ax[i] * dt;
    vy[i] := vy[i] + 0.5 * ay[i] * dt;
    ke := ke + 0.5 * (vx[i] * vx[i] + vy[i] * vy[i]);
    { writeln('ke = ',ke);}
    virial := virial + x[i] * ax[i] + y[i] * ay[i]
end
end;
{=====
=====}
procedure results (N, nave : integer;
Lx, Ly, dt : real;
var virial, ke, pe, xflux, yflux, time :
real);
var
pflux, pvirial, E, T : real;
begin
if time = 0.0 then
    writeln('vreme, T, E, pflux, pvirial');
    time := time + dt * nave;
    ke := ke / nave;
    pe := pe / nave;
    E := (pe + ke) / N; (* energia na 1 4actitsu *)
    T := ke / N; (* Temperture *)
    ke := 0.0; pe := 0.0;

```

```

    (* npivedennoe davlenie iz vychislenia potoka *)
    pflux := ((xflux/(2.0 * Lx)) + (yflux/ (2.0 * Ly)))/(dt * nave);
    xflux := 0.0;
    yflux := 0.0; (* npivedennoe davlenie iz viriala *)
    pvirial := (N * T)/(Lx * Ly) + 0.5 * virial/(Nave * Lx * Ly);
    virial := 0.0;
writeln(time : 9 : 3,    T : 15 : 4,    E : 15 : 4,    pflux : 15 : 4,
pvirial: 15: 4)
end;
{=====
=====}
procedure save_conf (var x,    y,    vx,    vy : component; N :
integer; Lx,    Ly :    real);
var fname :    string; dataout :    text; i :    integer;

begin
write('configuration file name... ');
readln( fname);
assign(dataout,    fname);
rewrite( dataout);
writeln(dataout,    N,    Lx,    Ly);

for i := 1 to N do
    writeln( dataout,    x[i],    y[i],    vx[i],    vy[i]);
    close( dataout)
end;

{=====
=====}
procedure centerMass(x,y:component; var xCentr, yCentr:real);
var
    i:integer;
begin
xCentr:=0;
yCentr:=0;
for i:=1 to N do begin
xCentr:=x[i]+xCentr;
yCentr:=y[i]+yCentr;
end;
xCentr:=xCentr/N;
yCentr:=yCentr/N;
writeln('xCentr = ',xCentr:8:2);
writeln('yCentr = ',yCentr:8:2);
readln;
end;

{=====
=====}

```

```

procedure funcRaspr (var x, y :component);
var
  i, j           : integer;
  dx, dy, r     : real;
  funcRasprFile : text ;
begin
assign(funcRasprFile, 'naborRasstoyanij');
rewrite(funcRasprFile);
for i := 1 to (N - 1) do
  for j := (i + 1) to N do begin
    dx := x[i] - x[j];
    dy := y[i] - y[j];
    separation(dx, dy, Lx, Ly);
    r := sqrt(dx * dx + dy * dy);
    writeln(funcRasprFile, r:7:3);
  end ;

close (funcRasprFile);
end;

{=====}
{===== program md2 (input, output);
{$N+}
const  c = 32768;                (* for random geration *)
       Nmax =100;                (* Maximum of particles *)
type
  component = array[1..Nmax] of real;
var
  x, y, vx, vy, ax, ay          : component;
  xCentr, yCentr                : real;
  N, nave, nset, iset, iave     : integer;
  Lx, Ly, dt, dt2               : real;
  virial, xflux, yflux, pe, ke, time : real;

{=====}
procedure initial (var x, y, vx, vy : component;
var N, nave, nset : integer; var Lx, Ly, dt, dt2 : real);
var irow, icol, nrow, ncol, i   : integer;
  ax, ay, xscale, yscale, Mx, My : real;
  vscale, vmax, vxcum, vycum    : real;
  fname                          : string;
  datain                         : text;
  newconf                        : char;
begin
write('number of molecules = ');
readln(N);
writeln('Box dimention');
write('Lx = ');
readln(Lx);
write('Ly = ');
readln(Ly);
write('Shag po vremeni = ');
readln(dt);
dt2 := dt * dt;
write('Shagov mezdu avereg, nave = ');
readln( nave);
write('Naborov usredneniya, nset = ');
readln(nset);

```

```

write('New config? (y/n)');
readln( newconf);
if (newconf = 'y') then
begin
(* na treugolnoj reshetke *)
vxcum := 0;
vycum := 0;
write('n molecul in row, nrow = ');
readln(nrow);
write('max skorost', vmax = ');
readln(vmax);
ncol := N div nrow;
(* rasstoyanie do bliжайshego soseda v kajdom napravlenii *)
ay := Ly / nrow;
ax := Lx / ncol;
i := 0;
for icol := 1 to ncol do
for irow := 1 to nrow do
begin
i := i + 1;
y[i] := ay * (irow - 0.5);
if ((irow mod 2) = 0) then
x[i] := ax * (icol - 0.25)
else
x[i] := ax * (icol - 0.75);
vx[i] := (2*random - 1) * vmax; (* v -random *)
vy[i] := (2*random - 1) * vmax;
(* sum Vx and Vy = 0 ? *)
vxcum := vxcum + vx[i];
vycum := vycum + vy[i];
write('x', i, '=', x[i]:6:3);
writeln(' y', i, '=', y[i]:6:3);
end;

readln;

vxcum := vxcum / N;
vycum := vycum / N;

for i := 1 to N do begin
vx[i] := vx[i] - vxcum;
vy[i] := vy[i] - vycum
end;
end

else (* old configur *)
begin
write('filename OLD configur, fname');
readln(fname);

{ write('otnositelnoe izmenenie skorost, vscale = '); }
{ readln(vscale); }
vscale := 1.0;

assign(datain, fname);
reset(datain);
readln(datain, N, Mx, My);
xscale := Lx / Mx;
yscale := Ly / My;

```

```

for i := 1 to N do begin

    readln(datain, x[i], y[i], vx[i], vy[i]);
    x[i] := x[i] * xscale;
    y[i] := y[i] * yscale;
    vx[i] := vx[i] * vscale;
    vy[i] := vy[i] * vscale end;
    close( datain)
end
end;

{=====
=====}
procedure periodic (var xtemp, ytemp, xflux, yflux : real; px,
py, Lx, LY : real);
begin
if xtemp < 0.0 then begin
    xtemp:= xtemp + Lx;
    xflux := xflux - px
end;

if xtemp > Lx then begin
    xtemp := xtemp - Lx;
    xflux := xflux + px
end;

if ytemp < 0.0 then begin
    ytemp := ytemp + Ly;
    yflux := yflux - py
end;

if ytemp > Ly then begin
    ytemp := ytemp - Ly;
    yflux := yflux + py
end
end;

{=====
=====}
procedure separation (var dx, dy : real; Lx, Ly : real);
begin
if abs(dx) > 0.5 * Lx then
    dx := dx * (1.0 - Lx / abs(dx));
if abs(dy) > 0.5 * Ly then
    dy := dy * (1.0 - Ly / abs(dy)) end;

{=====
=====}
procedure f (r : real; var force, potential : real);
var ri, ri3, ri6, g : extended;
begin
ri := 1.0 / r;
ri3 := ri * ri * ri;
ri6 := ri3 * ri3;
g := 24.0 * ri * ri6 * (2.0 * ri6 - 1.0);
{writeln('g=',g);}
force := g * ri;
potential := 4.0 * ri6 * (ri6 - 1.0)

```

```

end;
{=====}
=====}
procedure accel (var x, y, ax, ay : component; N : integer; Lx,
Ly : real; var pe : real);
var i, j : integer;
dx, dy, r, force, potential : real;
begin
for i := 1 to N do begin
ax[i] := 0.0;
ay[i] := 0.0
end;

for i := 1 to (N - 1) do
for j := (i + 1) to N do begin
dx := x[i] - x[j];
dy := y[i] - y[j];
separation(dx, dy, Lx, Ly);
r := sqrt(dx * dx + dy * dy);
f(r, force, potential);
ax[i] := ax[i] + force * dx;
ay[i] := ay[i] + force * dy;
ax[j] := ax[j] - force * dx;
ay[j] := ay[j] - force * dy;
pe := pe + potential
end
end;
{=====}
=====}
procedure Verlet (var x, y, vx, vy, ax, ay : component; N :
integer; Lx, Ly, dt, dt2 : real;
var virial, xflux, yflux, pe, ke : real);
var
i : integer; xnew, ynew : real;
begin
for i := 1 to N do begin
xnew := x[i] + vx[i] * dt + 0.5 * ax[i] * dt2;
ynew := y[i] + vy[i] * dt + 0.5 * ay[i] * dt2; (* sacTHMHO MSHseM
CKOpocrb, ncnoJib3yH crapoe ycKopeHHe *)
vx[i] := vx[i] + 0.5 * ax[i] * dt;
vy[i] := vy[i] + 0.5 * ay[i] * dt;
(* nepnoflMMeckHe xpaeebie ycnOBMS h BbmwciiHne noTOKa *)
periodic( xnew, ynew, xflux, yflux, vx[i], vy[i], Lx,
Ly);
x[i] := xnew; y[i] := ynew;
{ write('x',i,'=',x[i]:6:3);
write(' y',i,'=',y[i]:6:3);
readln; }
end;
accel(x, y, ax, ay, N, Lx, Ly, pe); (* pac4eT HOBoro ycKopenia
*)
{writeln('ke = ',ke);}
for i := 1 to N do (* OKOH^aTemHO MewieM CKOpocrt, wcnoribsya hobos
ycKopeHHe *) begin
vx[i] := vx[i] + 0.5 * ax[i] * dt;
vy[i] := vy[i] + 0.5 * ay[i] * dt;
ke := ke + 0.5 * (vx[i] * vx[i] + vy[i] * vy[i]);
{ writeln('ke = ',ke);}

```

```

        virial := virial + x[i] * ax[i] + y[i] * ay[i]
end
end;
{=====
=====}
procedure results (N,   nave : integer;
                  Lx,  Ly,  dt : real;
                  var virial, ke, pe, xflux, yflux, time :
real);
var
pflux,  pvirial,  E, T : real;
begin
if time = 0.0 then
    writeln('vreme, T, E, pflux, pvirial');
    time := time + dt * nave;
    ke := ke / nave;
    pe := pe / nave;
    E := (pe + ke) / N;      (* energia na 1 4actitsu *)
    T := ke / N;           (* Temperture *)
    ke := 0.0; pe := 0.0;
    (* npivedennoe davlenie iz vychislenia potoKa *)
    pflux := ((xflux/(2.0 * Lx)) + (yflux/ (2.0 * Ly)))/(dt * nave);
    xflux := 0.0;
    yflux := 0.0; (* npivedennoe davlenie iz viriala *)
    pvirial := (N * T)/(Lx * Ly) + 0.5 * virial/(Nave * Lx * Ly);
    virial := 0.0;
writeln(time : 9 : 3,   T : 15 : 4,   E : 15 : 4,   pflux : 15 : 4,
pvirial: 15: 4)
end;
{=====
=====}
procedure save_conf (var x,  y,  vx,  vy : component; N :
integer; Lx,  Ly : real);
var fname : string; dataout : text; i : integer;

begin
write('configuration file name... ');
readln( fname);
assign(dataout,  fname);
rewrite( dataout);
writeln(dataout,  N,  Lx,  Ly);

for i := 1 to N do
    writeln( dataout,  x[i],  y[i],  vx[i],  vy[i]);
    close( dataout)
end;

{=====
=====}
procedure centerMass(x,y:component; var xCentr, yCentr:real);
var
    i:integer;
begin
xCentr:=0;
yCentr:=0;
for i:=1 to N do begin
    xCentr:=x[i]+xCentr;
    yCentr:=y[i]+yCentr;

```

```

end;
xCentr:=xCentr/N;
yCentr:=yCentr/N;
writeln('xCentr = ',xCentr:8:2);
writeln('yCentr = ',yCentr:8:2);
readln;
end;

{=====}

procedure funcRaspr (var x, y :component);
var
  i, j           : integer;
  dx, dy, r      : real;
  funcRasprFile  : text ;
begin
assign(funcRasprFile, 'naborRasstoyanij');
rewrite(funcRasprFile);
for i := 1 to (N - 1) do
  for j := (i + 1) to N do begin
    dx := x[i] - x[j];
    dy := y[i] - y[j];
    separation(dx, dy, Lx, Ly);
    r := sqrt(dx * dx + dy * dy);
    writeln(funcRasprFile, r:7:3);
  end ;

close (funcRasprFile);
end;

{=====}

{=====}
{main}

begin

initial(x, y, vx, vy, N, nave, nset, Lx, Ly, dt, dt2);
pe := 0.0;
accel(x, y, ax, ay, N, Lx, Ly, pe);
time := 0.0;
pe := 0.0;
ke := 0.0;
xflux := 0.0;
yflux := 0.0;
virial := 0.0;

centerMass(x,y,xCentr, yCentr);
funcRaspr(x,y);

for iset := 1 to nset do begin
  for iave := 1 to nave do
    Verlet( x,y, vx,vy, ax,ay, N, Lx, Ly, dt, dt2, virial, xflux,
yflux, pe,ke);

```



```

    { results(N, nave, Lx, Ly, dt, virial, ke, pe, xflux, yflux,
time);}
    centerMass(x,y,xCentr, yCentr);

end;

writeln('s4as file budet perepisan');

writeln('save it if you need');

readln;

funcRaspr(x,y);

save_conf(x, y, vx, vy, N, Lx, Ly);

end.

=====}
{main}

begin

initial(x, y, vx, vy, N, nave, nset, Lx, Ly, dt, dt2);
pe := 0.0;
accel(x, y, ax, ay, N, Lx, Ly, pe);
time := 0.0;
pe := 0.0;
ke := 0.0;
xflux := 0.0;
yflux := 0.0;
virial := 0.0;

centerMass(x,y,xCentr, yCentr);
funcRaspr(x,y);

for iset := 1 to nset do begin
    for iave := 1 to nave do
        Verlet( x,y, vx,vy, ax,ay, N, Lx, Ly, dt, dt2, virial, xflux,
yflux, pe,ke);

    { results(N, nave, Lx, Ly, dt, virial, ke, pe, xflux, yflux,
time);}
    centerMass(x,y,xCentr, yCentr);

end;
writeln('s4as file budet perepisan');

writeln('save it if you need');
readln;
funcRaspr(x,y);
save_conf(x, y, vx, vy, N, Lx, Ly);
end.

```

## Пример выполнения проекта «Реализация интерфейса программы моделирования на основе молекулярной динамики»

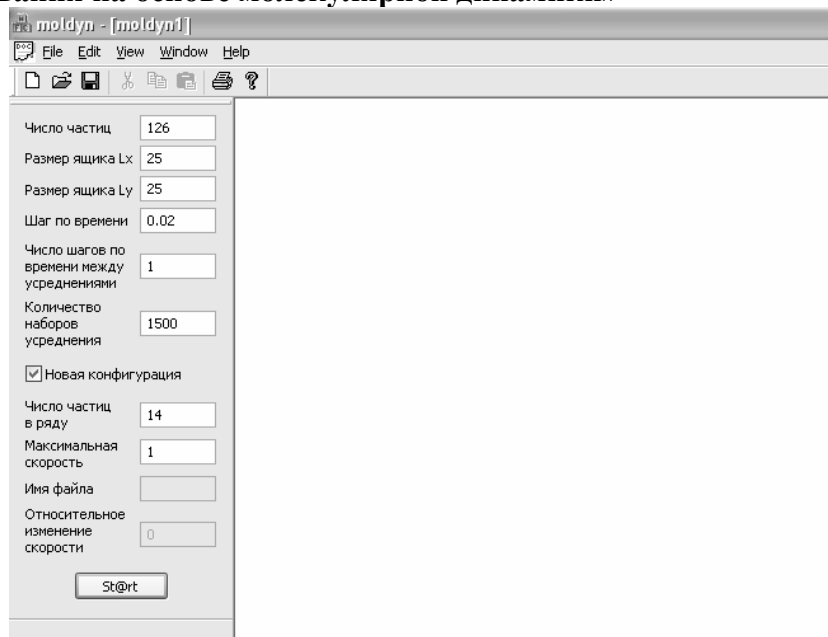


Рис. 3. Окно программы моделирования

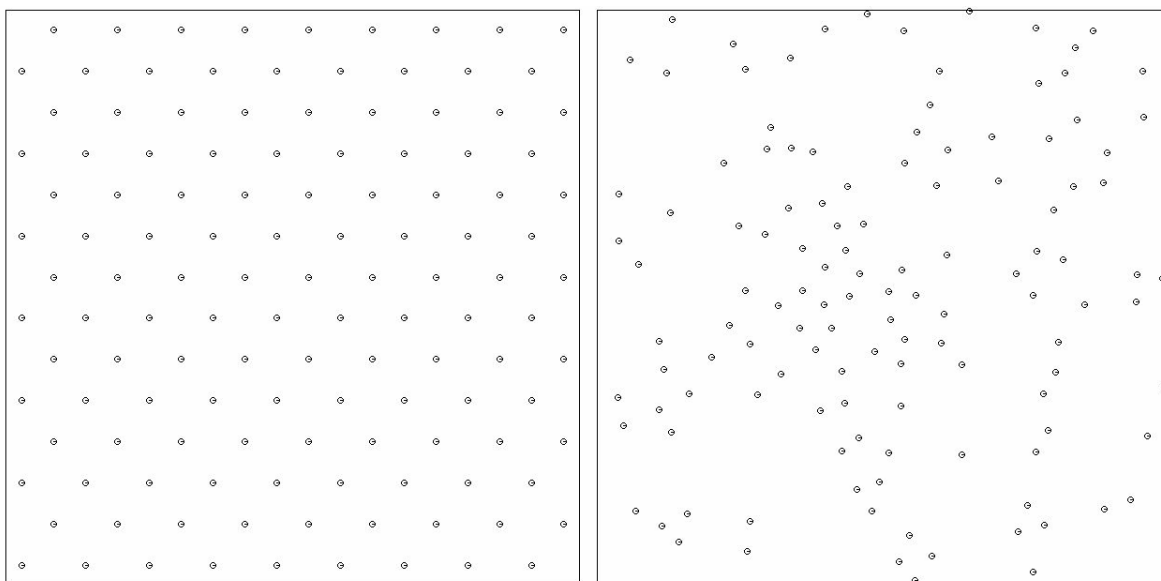


Рис. 4. Результат моделирования фазового перехода твердого тела в жидкость

## Список опечаток, исправлений и планируемых доработок

**#4 Спецкурс "Моделирование процессов и систем"** читается на факультете радиофизики и электроники для студентов 4-го курса, обучающихся по специальностям «Радиофизика» и «Физическая электроника».

Количество часов составляет 56, включая 24 лекционных, 26 лабораторных и 6 часов контролируемой самостоятельной работы.

Данный специальный курс состоит из **следующих составных частей:**

Введение (цели, задачи, методы)

I. Моделирование динамических систем

II. Объектно-ориентированный анализ и моделирование систем

III. Инструментальные средства моделирования

IV. Модели фотофизических процессов

V. Примеры моделей систем (включая моделирование электронных приборов)

При изучении данного специального курса используются и развиваются навыки, полученные при изучении общих курсов («Молекулярная физика», «Электричество», «Оптика», «Атомная физика», "Математическое моделирование", "Методы вычислительного эксперимента", «Программирование», «Теория колебаний»).

**Нумерация рисунков и литературы– в пределах лекции (не сквозная) !  
Проверить нумерацию рисунков в лекциях 3 (2->3) и 10 (9->10->11)**

### Источники информации по MPI

1. Сайт <http://www.mcs.anl.gov/mpi> (Argonne National Laboratory, США)
2. Сайт <http://www.parallel.ru> (Научно-исследовательский ВЦ МГУ)
3. Сайт <http://www.cluster.bsu.by> (Белгосуниверситет, Минск)
4. В.В.Воеводин, Вл.В.Воеводин. Параллельные вычисления. – СПб.: БХВ-Петербург, 2002. – 609 с.
5. Шпаковский Г.И. Организация параллельных ЭВМ и суперскалярных процессоров: Учеб. пособие. Мн.: Белгосуниверситет, 1996. 284 с.
6. Буза М.К. Введение в архитектуру компьютеров: Учеб. пособие. Мн.: БГУ, 2000. 253 с.
7. Сайт <http://www.top500.org>
8. Г.И.Шпаковский, Н.В.Серикова. Программирование для многопроцессорных систем в стандарте MPI. – Мн.:БГУ, 2002. – 323 с.
9. Лацис А.О. Как построить и использовать суперкомпьютер. – М.:Бестселлер. 2003. – 240 с.
10. В.Г.Олифер, Н.А.Олифер. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. 2-е изд. – СПб.:Питер, 2003. –864 с.
11. В.В.Корнеев. Параллельные вычислительные системы. – М.: “Нолидж”, 1999. – 320 с.
12. В.П.Гергель. Оценка эффективности параллельных вычислений для Intel-процессорных вычислительных кластеров. Материалы международного научно-практического семинара “Высокопроизводительные параллельные вычисления на кластерных системах”./Под ред. проф. Р.Г.Стронгина. Нижний Новгород: Изд-во Нижегородского университета, 2002.
13. С.А.Немнюгин, О.Л.Стефик. Параллельное программирование для многопроцессорных вычислительных систем.–СПб.:Петербург, 2002.– 400 с.