Министерство образования и науки Российской Федерации

САНКТ–ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Приоритетный национальный проект «Образование»
Национальный исследовательский университет

Ю. Б. СЕНИЧЕНКОВ

МОДЕЛИРОВАНИЕ

КОМПЬЮТЕРНЫЙ ПРАКТИКУМ

Рекомендовано Учебно-методическим объединением по университетскому политехническому образованию в качестве учебного пособия для студентов высших учебных заведений, обучающихся по направлению подготовки магистров «Системный анализ и управление»

Санкт-Петербург
Издательство Политехнического университета
2013

УДК 681.3.06
ББК 32.693.26
К 60

Рецензенты:

Доктор педагогических наук, профессор
СПбГЭТУ «ЛЭТИ»
С. Н. Поздняков
Кандидат физико-математических наук,
доцент СПбГПУ
С. П. Воскобойников

Сениченков Ю. Б. **Моделирование систем. Компьютерный практикум:** учеб. пособие / Ю. Б. Сениченков. – СПб.: Изд-во Политехнического ун-та, 2012. – 189 с.

Пособие представляет собой сборник учебных заданий, методических указаний, и руководств пользователей программных сред, с помощью которых предполагается выполнение заданий. Пособие содержит последовательно усложняющиеся задания для учебной дисциплины «Моделирование», состоящей из разделов «Моделирование динамических систем» и «Компонентное моделирование сложных динамических систем», и связанных с ней дисциплин. Все материалы объединены под единым названием «Моделирование. Компьютерный практикум».

Пособие предназначено для студентов высших учебных заведений, обучающихся по направлению подготовки магистров «Системный анализ и управление». Пособие может быть также использовано при подготовке магистров по направлениям «Информатика и вычислительная техника», «Фундаментальная информатика и информационные технологии». Пособие может быть полезно и для инженеров, занимающихся проектированием и исследованием сложных динамических систем. Оно может быть использовано при обучении в системах повышения квалификации, в учреждениях дополнительного профессионального образования.

Работа выполнена в рамках реализации программы развития национального исследовательского университета «Модернизация и развитие политехнического университета как университета нового типа, интегрирующего мультидисциплинарные научные исследования и надотраслевые технологии мирового уровня с целью повышения конкурентоспособности национальной экономики»

Печатается по решению редакционно-издательского совета
Санкт-Петербургского государственного политехнического университета.

ISBN

Сениченков Юрий Борисович

МОДЕЛИРОВАНИЕ
КОМПЬЮТЕРНЫЙ ПРАКТИКУМ

Учебное пособие

Налоговая льгота – Общероссийский классификатор продукции
ОК 005-93, т. 2; 953005 – учебная литература

Подписано к печати XX.XX.2013. Формат 60x84/16. Печать цифровая.
Усл. печ. л. 12,0. Тираж 40 экз. Заказ

Отпечатано с готового оригинал-макета, предоставленного автором,
в Издательстве Политехнического университета.
195251, Санкт-Петербург, Политехническая ул., 29.
Тел.: (812) 550-40-14
Тел./факс: (812) 297-57-76

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Современное математическое моделирование применяется в различных областях. Используемые технологии моделирования определяются, прежде всего, типами математических моделей. Мы будем говорить только о динамических системах и их обобщении — гибридных системах.

Дисциплина «Моделирование» для студентов направления «Фундаментальная информатика и информационные технологии» может быть представлена как совокупность тем, посвященных основам моделирования и современным технологиям компьютерного моделирования. Это — «Моделирование динамических систем», «Компонентное моделирование динамических систем», «Технологии вычислений» и «Визуальные среды моделирования сложных динамических систем».

Изучение современных технологий моделирования немыслимо без практической и самостоятельной работы. Практическая работа студентов проходит в форме лабораторных и курсовой работы. Студенты не только выполняют учебные задания, но и самостоятельно осваивают современное программное обеспечение. В основе учебных заданий лежат задачи из хорошо известных задачников и учебников по механике, электричеству. Эти заимствования сделаны сознательно, так нам важно не просто решить задачу (это уже делалось при изучении соответствующей дисциплины), а решить хорошо известную задачу с помощью различных средств визуального моделирования.

Современные программные среды моделирования — это сложно организованные программные комплексы, освоить которые достаточно трудно: справочные системы сред моделирования представляют собой многостраничные тексты, написанные для профессиональных пользователей, да еще и чаще всего на английском языке.

Преодолеть эту трудность проще всего с помощью специальных методических указаний, написанных преподавателем, представляющих собой краткие руководства пользователя, решающего только

определенные задачи. Такие руководства были разработаны для всех программных сред, использующихся в дисциплине «Моделирование».

Более сложным и трудоемким, однако, и более эффективным, по мнению автора, является путь, связанный с проведением семинарских занятий, на которых студенты представляют и учат своих товарищей применять изучаемые программные среды. С этой целью мы предлагаем организовать изучение темы «Визуальные среды моделирования сложных динамических систем» в форме семинара.

У семинара, с методической точки зрения, несколько целей:

•Научить студента самостоятельно работать с технической документацией на русском и английском языках при подготовке к выступлениям. С одной стороны, многие нужные руководства еще не переведены на русский, с другой стороны, часть из них нет необходимости переводить вообще — студент университета должен владеть хотя бы «техническим» английским.

•Помочь студенту научиться выступать на семинарах и вести диалог с коллегами: уметь классифицировать программные средства (опираясь на теорию), видеть их достоинства и недостатки (самостоятельно работая со средой), подбирать иллюстративные примеры.

•Развить навыки сознательного выбора и использования программных сред для решения практических задач.

За время преподавания дисциплины «Моделирование» было накоплено большое число примеров, положенных в основу лабораторных и курсовых работ. Как показал опыт, выполнение лабораторных работ лучше всего проводить следующим образом. Сначала выдаются только задания. По ходу выполнения задания конкретным студентом, практически всегда становится ясным, кто сможет выполнить работу самостоятельно, а кому нужны детальные разъяснения. На это случай практически по каждому из разделов были подготовлены методические указания. Если студент не выполняет задание в срок, ему «открывается» доступ к методическим материалам. Выдавать задания вместе с методическими указаниями рекомендуется в том случае, если программа курса не предусматривает достаточно часов на практику и самостоятельную работу.

1. ОДНОКОМПОНЕНТНЫЕ ДИНАМИЧЕСКИЕ СИСТЕМЫ

Задания этого раздела относятся к части «Моделирование динамических систем». Название раздела подчеркивает, что речь идет об изолированных системах, «замкнутых» системах дифференциальных или разностных уравнений. Иными словами, эта система подготовлена заранее, представляет собой модель в окончательном виде, и решение этой системы может быть найдено с помощью программного обеспечения, предназначенного для моделирования изолированных систем — Matlab, Rand Model Designer (интерфейсы «непрерывная» и «гибридная системы), Maple, Mathematica, Sage. Пакеты Matlab, Rand Model Designer используются в лабораторных работах для получения численного решения, а последние три (в зависимости от наличия лицензии, достаточно одного) — для получения символьного решения. Символьное решение используется как тестовое для проверки полученного численного решения.

«Задание 1» и «Задание 2» предназначены, в основном, для самостоятельного изучения языков пакетов, используемых для лабораторных работ. Дифференциальные и разностные уравнения и методы их решения уже изучались студентами, поэтому основное внимание уделяется использованию программного обеспечения.

«Задание 3» и «Задание 4» иллюстрируют основные понятия качественной теории дифференциальных уравнений. Сначала исследуется нелинейное дифференциальное уравнение первого порядка, а затем линейные системы на плоскости.

«Задание 5» относится к разделу «Устойчивость» и позволяет сравнить поведение исходной и линеаризованной систем в окрестности особой точки.

«Задание 6» предназначено для углубленного изучения возможностей языков программирования, используемых пакетов.

ЗАДАНИЕ 1. ПОСТЕЙШАЯ ДИНАМИЧЕСКАЯ СИСТЕМА

1. Решить численно уравнения:

$$\frac{dx}{dt} = -x + (a_2 \cdot t^2 + a \cdot_1 t + a_0); \quad t \in [-T, T], \quad x(0) = x_0, \qquad (1.1)$$

$$\frac{dy}{dt} = -y + (a_2 \cdot \cos(t) + a_1 \cdot \sin(t) + a_0); \; t \in [-T, T], \; y(0) = y_0. \qquad (1.2)$$

Коэффициенты уравнения для конкретного варианта приведены в табл. 1.1.

Таблица 1.1

Коэффициенты уравнений

N	a_2	a_1	a_0	***T***	x_0	**Уравнение**
1	1	0	0	1	–6	1,1
2	0	1	0	2	–5	1,1
3	1	1	0	3	–4	1,1
4	0	1	1	4	–3	1,1
5	1	0	1	1	–2	1,1
6	1	1	1	2	–1	1,1
7	1	0	0	3	1	1,2
8	0	1	0	4	2	1,2
9	1	1	0	1	3	1,2
10	0	1	1	2	4	1,2
11	1	0	1	3	5	1,2
12	1	1	1	4	6	1,2

Нарисовать графики построенных решений.

2. Найти решение приведенной системы:

- используя формулу

$$\frac{dx}{dt} + f(t) \cdot x = g(t);$$
$$x(0) = x_0;$$
$$F(t) = \int_0^t f(t)dt;$$
$$x(t) = e^{-F(t)} \cdot (x_0 + \int_0^t g(t) \cdot e^{F(t)} dt);$$

- как сумму решений — решения однородного уравнения и частного решения неоднородного уравнения.

3. Построить график относительной погрешности численного решения, сравнивая численное решение с найденным в явном виде решением.

Указания. Прежде всего, рекомендуется найти решение вручную, а затем используя математический пакет, например Maple. Найденные решения использовать для проверки правильности численных решений.

ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Рассмотрим дифференциальное уравнение первого порядка относительно вещественной переменной x с постоянными коэффициентами:

$$\frac{dx}{dt} = a \cdot x + b;$$
$$x(0) = x_0$$

и его решение:

$$x(t) = e^{at} \cdot x_0 + \frac{b}{a}(e^{at} - 1).$$

Требуется построить численное решение, сравнить его с аналитическим (символьным), построить графики абсолютной и относительной ошибок.

Создадим проект в среде MvStudium (Rand Model Designer. Для решения таких простых задач можно воспользоваться любой средой из семейства MvStudium, см. http://www.mvstudium.com).

Прежде всего, создадим новый проект (рис. 1.1). Выберем команду «Проект»/«Новый». Как видно из рис. 1.1, диалог появился на фоне старого проекта. Наша система изолированная, динамическая, поэтому выбираем тип модели «Непрерывный элементарный объект».

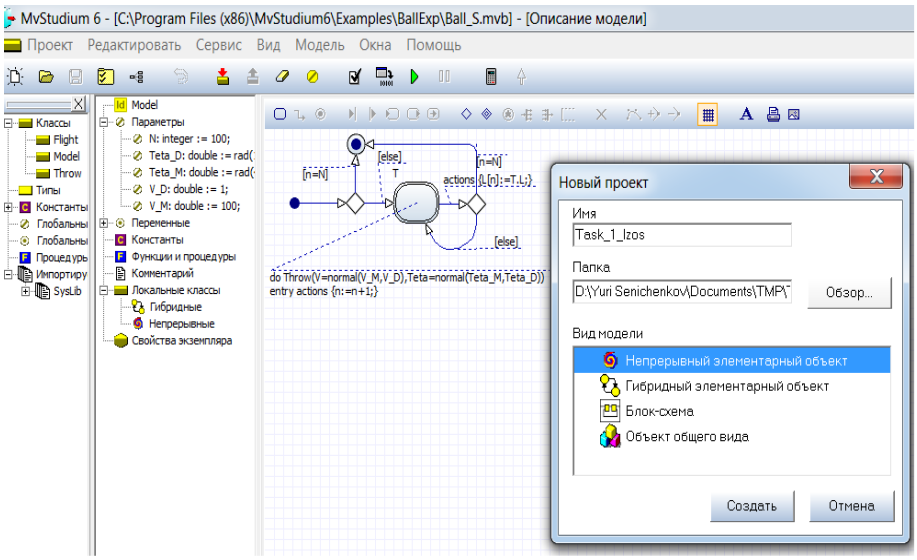


Рис. 1.1. Выбор типа модели для динамической системы, заданной дифференциальным уравнением. Команда «Сервис»/«Настройки» на фоне старого проекта

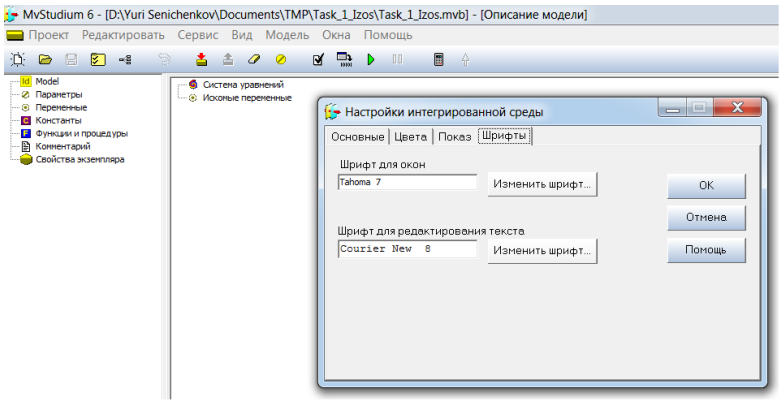


Рис. 1.2. Первоначальный вид главного окна нового проекта. Дополнительно выполнена команда «Сервис»/«Настройки»

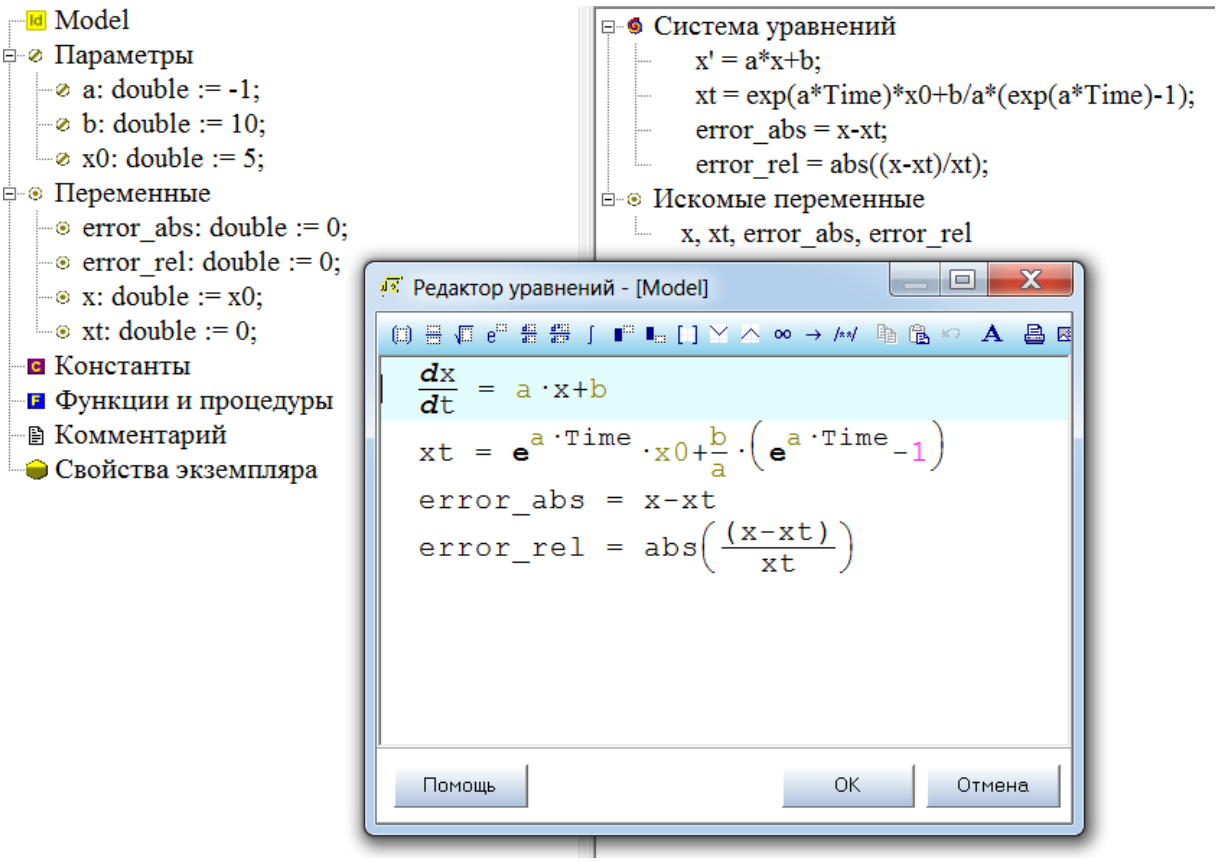


Рис. 1.3. Уравнение, которое предстоит решить среде, и его контрольное решение. Окно «Редактор Уравнений» на фоне главного окна

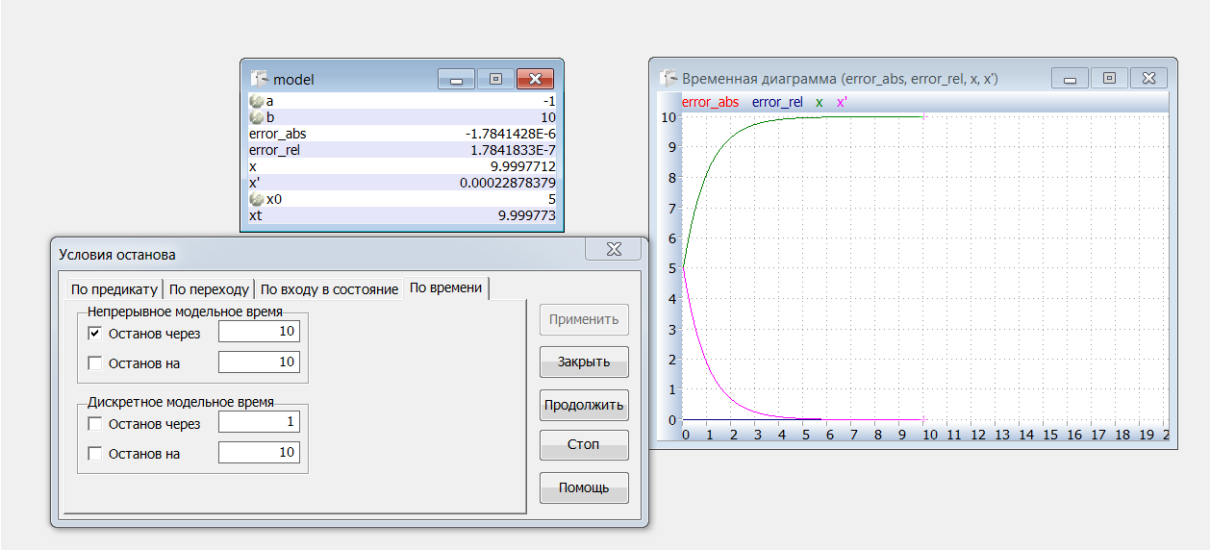


Рис. 1.4. Численное решение на промежутке [0,10]

На рис. 1.2 показано, что увидит пользователь, создавший новый проект. Дополнительно показан диалог, позволяющий изменить часть параметров отображаемых окон. На рис. 1.3 показано, как выглядит окно редактора уравнений, приведено описание всех нужных переменных. Рис. 1.4 содержит графики: численное решение уравнения, контрольное решение, ошибки.

В задании сказано, что решение нужно получить на промежутке [-10,10]. Системная переменная «Time» изменяется, начиная с нуля. Введем новое время «Time1» (рис. 1.5) и сместим начало отсчета. Мы сместили отсчет на одну единицу. Смещение начала отсчета потребует внести изменения в формулу точного решения. В результате получим правильное решение на новом промежутке (рис. 1.6).

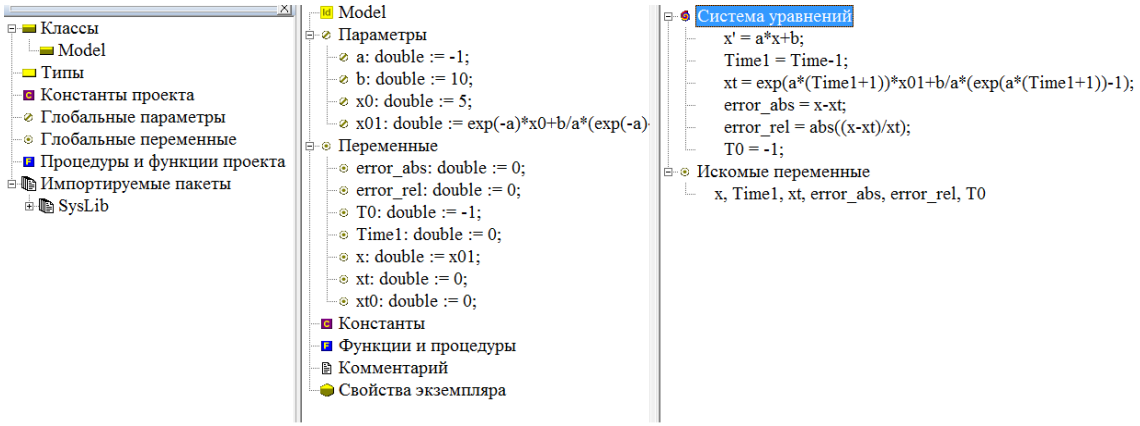


Рис. 1.5. Прием, позволяющий сместить начальную точку отсчета времени

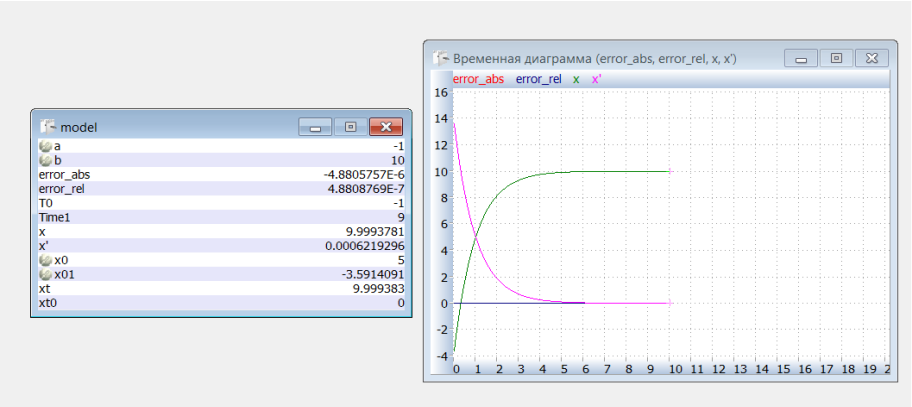


Рис. 1.6. Решение на промежутке [-1,9]

ЗАДАНИЕ 2. ДИНАМИЧЕСКИЕ СИСТЕМЫ. ОТОБРАЖЕНИЯ

Реализовать следующие отображения [1].

1. Отображение Хенона

$$\begin{cases} x_{n+1} = -\left(y_n - x_n^2\right)\cdot \sin\left(\dfrac{2\pi}{p_1}\right) + x_n \cdot \cos\left(\dfrac{2\pi}{p_1}\right) \\ x_{n+1} = \left(y_n - x_n^2\right)\cdot \cos\left(\dfrac{2\pi}{p_1}\right) + x_n \cdot \sin\left(\dfrac{2\pi}{p_1}\right) \end{cases};$$

рекомендуемые значения $p_1 = 6$.

2. Отображение Чирикова

$$\begin{cases} x_{n+1} = y_{n+1} + x_n \\ y_{n+1} = y_n + p_1 \sin(x_n) \end{cases};$$

рекомендуемые значения $p_1 = 0{,}7$.

3. Отображение Мира

а)
$$\begin{cases} x_{n+1} = p_1 x_n + y_n \\ y_{n+1} = p_2 + x_n^2 \end{cases};$$

рекомендуемые значения $p_1 = 1; p_2 = -0{,}5952$.

б)
$$\begin{cases} x_{n+1} = (1 - p_1)x_n + y_n \\ y_{n+1} = p_1 p_2 x_n + p_3 y_n - p_1 x_n^3 \end{cases};$$

рекомендуемые значения $p_1 = 2{,}68; p_2 = -0{,}1; p_3 = 0{,}9$.

в)
$$\begin{cases} x_{n+1} = p_2 y_n - F(x_n) \\ y_{n+1} = -x_n + F(x_{n+1}) \end{cases}$$

$$F(x) = p_1 x + (1 - p_1)\frac{2x^2}{1 + x^2};$$

рекомендуемые значения $p_1 = -0{,}5; p_2 = 1$.

г)
$$\begin{cases} x_{n+1} = x_n^2 + p_1 x_n - y_n^2 + p_2 \\ y_{n+1} = 2x_n y_n - \dfrac{5}{2} p_1 y_n \end{cases};$$

рекомендуемые значения $p_1 = -0{,}95126; p_2 = -0{,}8788$.

4. Отображение Катала

$$\begin{cases} x_{n+1} = p_1 x_n + y_n \\ y_{n+1} = p_2 + x_n^2 \end{cases};$$

рекомендуемые значения $p_1 = 0{,}7; p_2 = -0{,}82.$

5. Отображение Заславского

$$\begin{cases} x_{n+1} = x_n \cos(p_1) - (y_n + p_2 \sin(x_n)) \cdot \sin(p_1) \\ y_{n+1} = x_n \sin(p_1) + (y_n + p_2 \sin(x_n)) \cdot \cos(p_1) \end{cases};$$

рекомендуемые значения $p_1 = 0{,}8; p_2 = 1.$

6. Отображение Улама

$$\begin{cases} x_{n+1} = x_n + \cos(2\pi y_n) \\ y_{n+1} = y_n + \dfrac{p_1}{x_{n+1}} \end{cases};$$

рекомендуемые значения $p_1 = 5$ или $p_1 = 20.$

7. Гравитационное отображение Заславского

$$\begin{cases} x_{n+1} = x_n + \cos(2\pi y_n) \\ y_{n+1} = y_n + p_1 x_{n+1} \end{cases}.$$

8. Фрактал Жюлиа

$$\begin{cases} x_{n+1} = x^2{}_n - y^2{}_n + p_1 \\ y_{n+1} = 2 y_n x_n + p_2 \end{cases};$$

рекомендуемые значения $p_1 = -0{,}22; p_2 = -0{,}74.$

9. Фрактал Жюлиа-2

$$\begin{cases} x_{n+1} = \dfrac{1}{2}(x_n + x_n^3 - 3 x_n y_n^2) \\ y_{n+1} = \dfrac{1}{2}(y_n + y_n^3 - 3 y_n x_n^2) \end{cases}.$$

10. Фрактал Жюлиа-4

$$\begin{cases} x_{n+1} = (5x_n - 2x_n^3 + 6x_n y_n^2)/3 \\ y_{n+1} = (5y_n + 2y_n^3 - 6y_n x_n^2)/3 \end{cases}.$$

ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Рассмотрим скалярное разностное уравнение первого порядка, λ — параметр:

$$x_{n+1} = \lambda \cdot x_n \cdot (1 - x_n), n = 0,1...;$$
$$x_0 = x0$$

и попробуем найти его решение. Казалось бы, семейство визуальных сред MVStudium не приспособлено для решения таких задач. Но ведь это — динамическая система, только время дискретное!

Создадим новый проект, но только теперь выберем тип «Гибридная система». Визуальное представление называется гибридным автоматом. Посмотрите на рис. 1.7 — на нем действительно изображен автомат, который через секунду модельного времени покидает текущее состояние S1 и возвращается в него же.

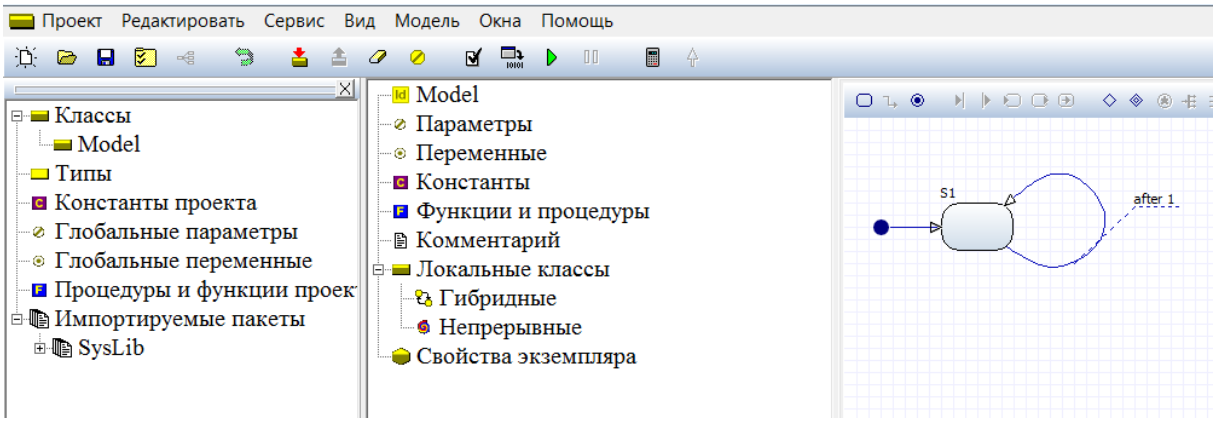


Рис. 1.7. Гибридный автомат, циклическое возвращение в состояние S1

При каждом возвращении к текущему состоянию вычислим очередное x_{n+1}, как того требует уравнение. Обозначим очередное значение через zn (рис. 1.8).

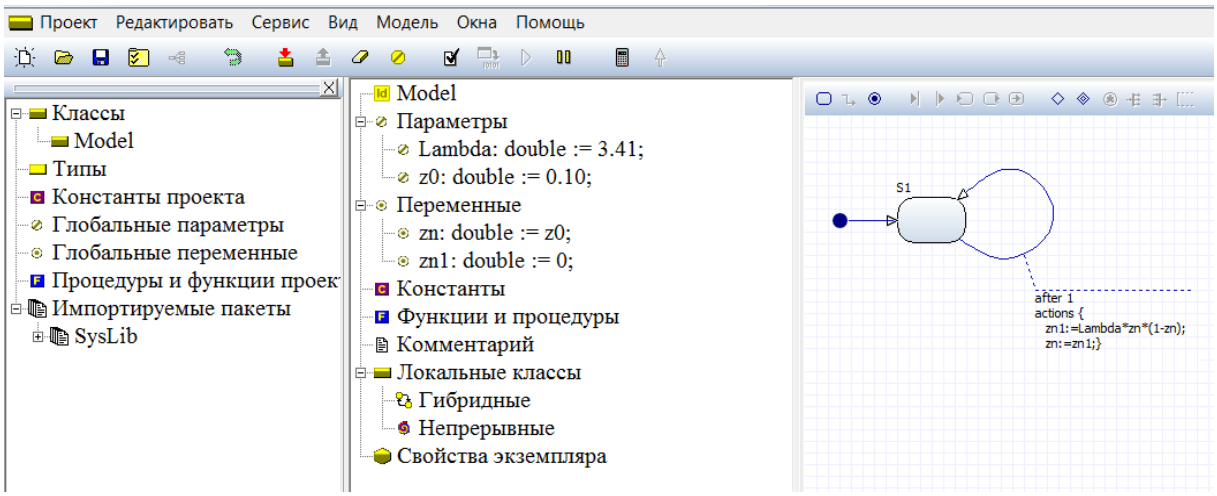


Рис. 1.8. При каждом возвращении в состояние S1 вычисляется очередное значение последовательности zn и сохраняется предыдущее значение, так как разностное уравнение первого порядка

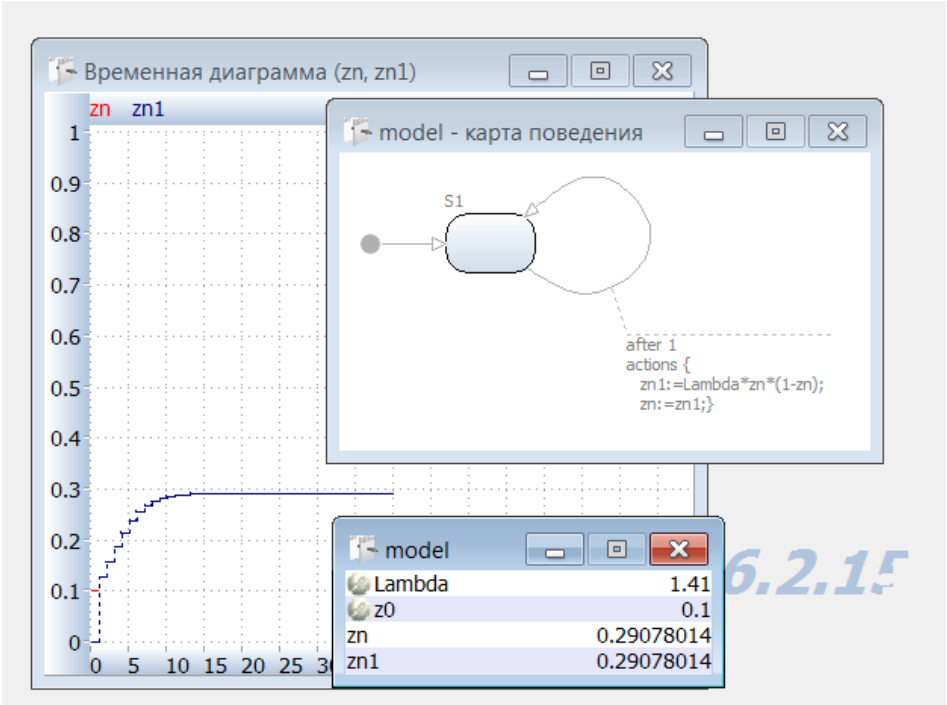


Рис. 1.9. Интуитивно ожидаемое решение разностного уравнения, трактуемого как уравнение численности некоторой популяции. С ростом времени численность популяции стабилизируется

Проведем два численных эксперимента. Выберем $\lambda = 1{,}41$ (рис. 1.9) и $\lambda = 3{,}41$ (рис. 1.10). Как видно на рис. 1.10, с увеличением значения параметра появляются колебания.

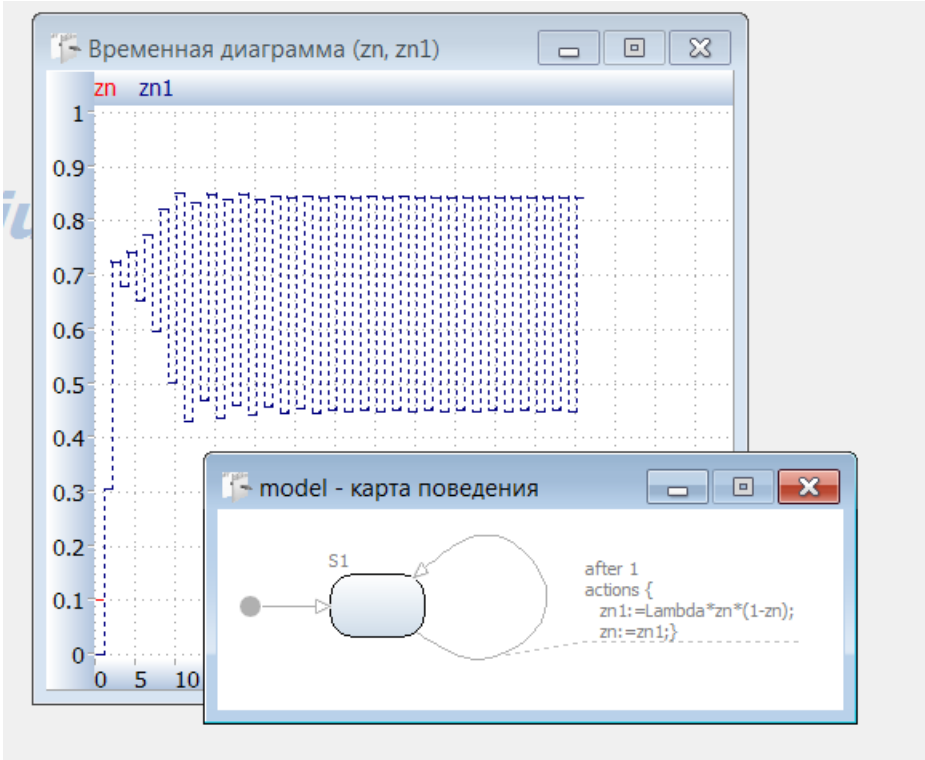


Рис. 1.10. Появление колебаний при увеличении значения параметра

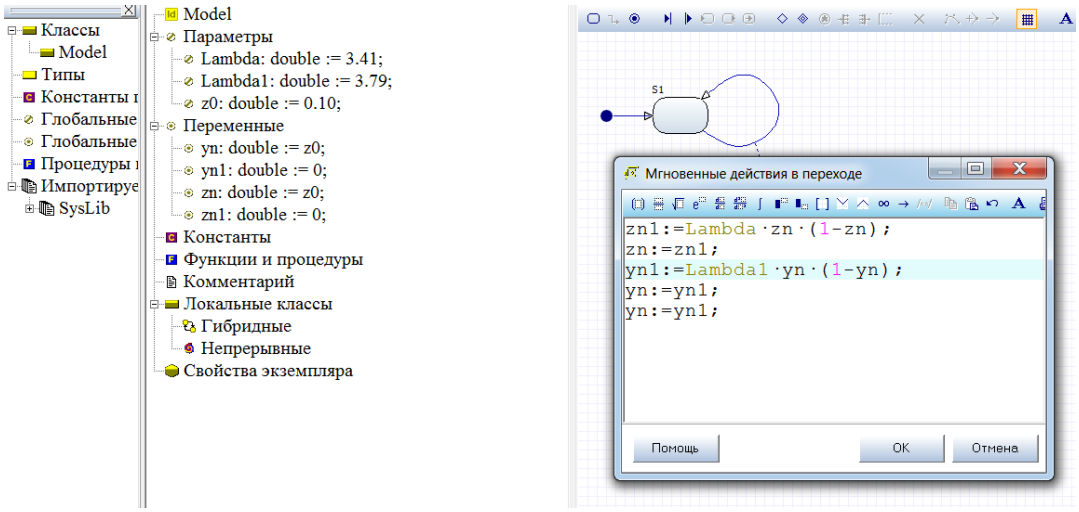


Рис. 1.11. Решение системы разностных уравнений

В исходных учебных заданиях — два разностных уравнения, иными словами, система. У нас — одно уравнение. Используем рассматриваемый пример для того чтобы показать, как решить систему. Добавляем еще одно разностное уравнение (переменная yn) со своим параметром (рис. 1.11).

На рис. 1.12 (правая часть) показано, как можно настраивать окно «Диаграмма». В таблице «Список переменных» последняя колонка помечена символом X («крестик»). Выберите нужную переменную из списка и поставьте «крестик» в последней колонке (щелчком соответствующей кнопки мыши). Помеченная переменная «станет» осью абсцисс, и вы получите вместо временной диаграммы — фазовую.

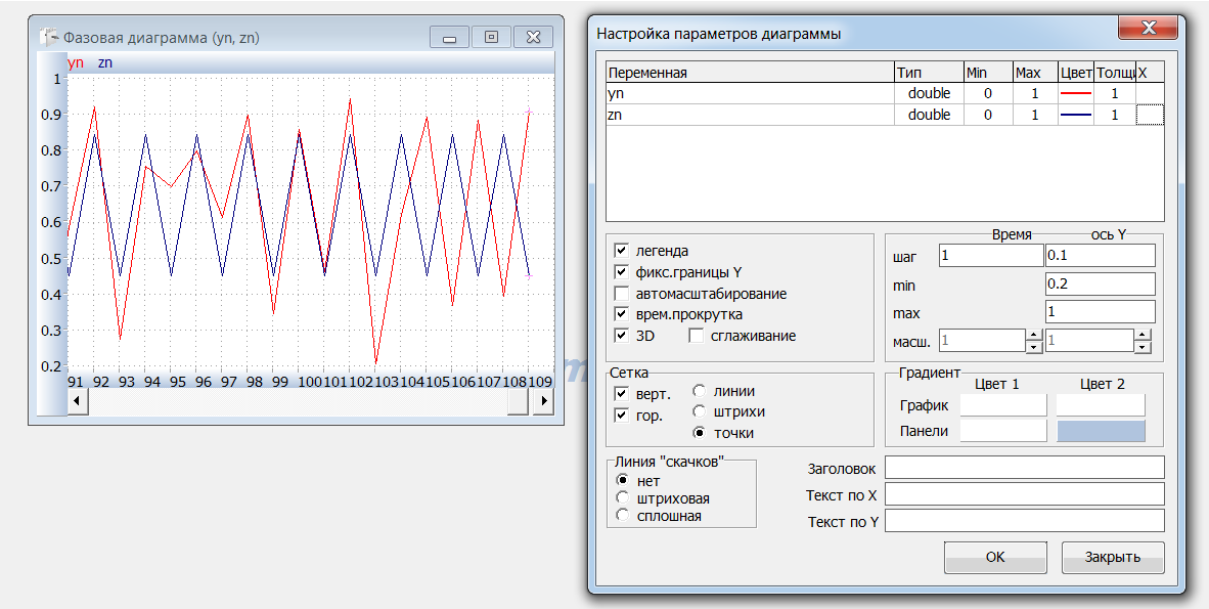


Рис. 1.12. Решение двух разностных уравнений
и панель настройки параметров диаграммы

ЗАДАНИЕ 3. ОСОБЫЕ ТОЧКИ

1. Построить графики правых частей дифференциальных уравнений из табл. 1.2 как функций от x и пометить особые точки на промежутке $[-2\pi; 2\pi]$. Определить, какие из них устойчивы, а какие — нет. Построить фазовый портрет.

2. Написать программу поиска корней функции одной переменной на языках Matlab и MVL. Использовать локальные и глобальные методы поиска: на первом этапе методы деления отрезка пополам, метод золотого сечения, случайный поиск, на втором этапе — локальные методы:

- метод ложного положения

$$x_{n+1} = x_n - f(x_n) \cdot \frac{x_n - x_0}{f(x_n) - f(x_0)};$$

- метод секущих

$$x_{n+1} = x_n - f(x_n) \cdot \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})};$$

- метод Ньютона

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Сравнить вычислительные затраты методов. Процедуры должны иметь те же параметры, что и процедура ZEROIN.

Таблица 1.2

Уравнения для построения графиков

№ п/п	Уравнение	№ п/п	Уравнение	№ п/п	Уравнение
1	$\frac{dx}{dt} = x^3 - x$	5	$\frac{dx}{dt} = \sin x - x^2$	9	$\frac{dx}{dt} = x^3 - \sin x$
2	$\frac{dx}{dt} = \frac{1}{2}x^2 + 2x - 1$	6	$\frac{dx}{dt} = e^{-x} - \sin x$	10	$\frac{dx}{dt} = e^{-x} - x^3$
3	$\frac{dx}{dt} = \sin x + x$	7	$\frac{dx}{dt} = e^{-x} + \cos x$	11	$\frac{dx}{dt} = e^{-x} - x^2 + 2$
4	$\frac{dx}{dt} = \sin x + x^2$	8	$\frac{dx}{dt} = x^4 - x^2$	12	$\frac{dx}{dt} = \cos x + x^2$

Учебные программы поиска корней функции одной переменной строятся в соответствии с табл. 1.3.

В случайном поиске очередная точка деления отрезка находится случайным образом (равномерное распределение).

Таблица 1.3

Сочетание глобальных и локальных методов в учебной программе

Глобальный метод	Локальный метод
1. Метод деления отрезка пополам Matlab	Метод Ньютона RMD
2. Метод золотого сечения Matlab	Метод секущих RMD
3. Случайный поиск Matlab	Метод ложного положения RMD
4. Метод деления отрезка пополам Matlab	Метод ложного положения RMD
5. Метод золотого сечения Matlab	Метод секущих RMD
6. Случайный поиск Matlab	Метод Ньютона RMD
7. Метод деления отрезка пополам RMD	Метод Ньютона Matlab
8. Метод золотого сечения RMD	Метод секущих Matlab
9. Случайный поиск RMD	Метод ложного положения Matlab
10. Метод деления отрезка пополам RMD	Метод ложного положения Matlab
11. Метод золотого сечения RMD	Метод секущих Matlab
12. Случайный поиск RMD	Метод Ньютона Matlab

Сравнить найденные значения со значениями, полученными с помощью программ пакетов (Matlab, RMD) поиска корней вещественной функции одной переменной.

ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Рассмотрим дифференциальное уравнение первого порядка (рис. 1.13). Сначала построим график правой части, как функции от x. Для этого закомментируем строчки, определяющие дифференциальное уравнение. Мы должны увидеть, и видим, что существуют три корня (рис. 1.14).

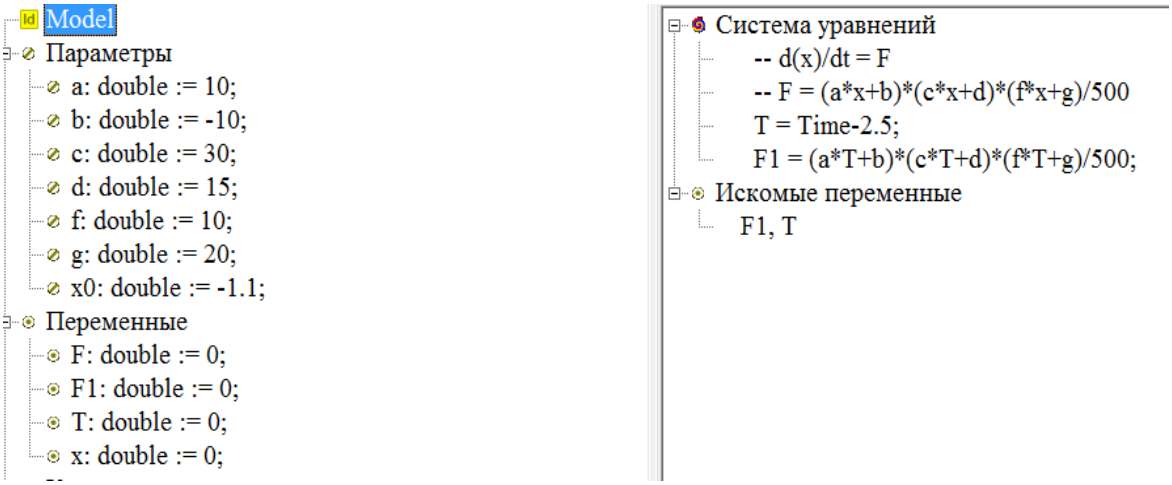


Рис. 1.13. Дифференциальное уравнение с кубическим полиномом в правой части

На рис. 1.14 мы видим график функции и окно, определяющие условие останова. Параметры выбраны так, чтобы показать значение функции только в окрестности корней. Обратите внимание на необычные «характеристики» времени, используемого в среде RMD, речь идет о непрерывном компоненте времени и дискретном. Сейчас мы рассматриваем динамическую систему с непрерывным временем.

Раскомментируем строчки, соответствующие дифференциальному уравнению, и посмотрим на решение дифференциального уравнения при заданных начальных условиях (рис. 1.15). По мере того, как x приближается к значению (-1/2), правая часть стремится к нулю. Как это связано с корнями правой части?

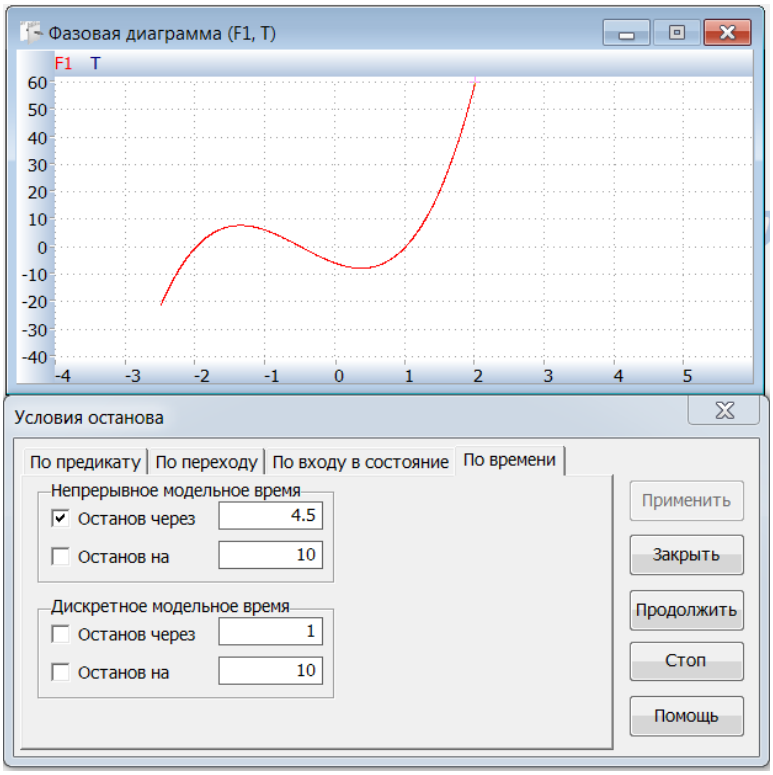


Рис. 1.14. График функции

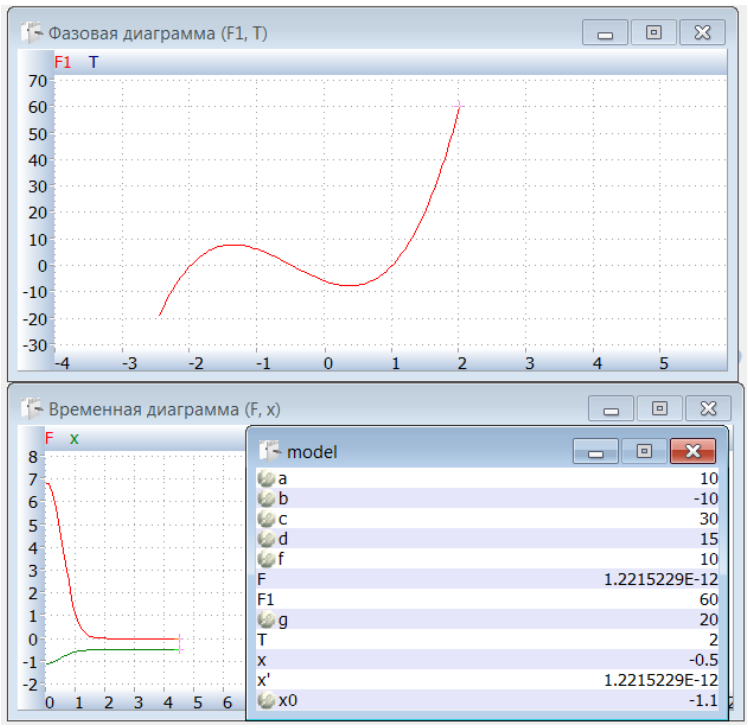


Рис. 1.15. Решение уравнения
и график правой части

ЗАДАНИЕ 4. ДИНАМИЧЕСКИЕ СИСТЕМЫ НА ПЛОСКОСТИ

Построить решение и численное решение системы

$$\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x} + \mathbf{b}; \mathbf{x}, \mathbf{b} \in \Re^2,$$

нарисовать их, используя пакет RMD, в виде временной и фазовой диаграммы, предварительно построив матрицу $\mathbf{A}$ по формулам:

$$\mathbf{A} = \mathbf{S} \cdot \begin{bmatrix} \lambda & 0 \\ 0 & \mu \end{bmatrix} \cdot \mathbf{S}^{-1}; \ \mathbf{A} = \mathbf{S} \cdot \begin{bmatrix} \lambda & 1 \\ 0 & \lambda \end{bmatrix} \cdot \mathbf{S}^{-1}; \ \mathbf{A} = \mathbf{S} \cdot \begin{bmatrix} \mu & \nu \\ -\nu & \mu \end{bmatrix} \cdot \mathbf{S}^{-1}.$$

Таблица 1.4

Тип особой точки и вид преобразования

Вариант	**Тип особой точки**	**Преобразование подобия**
1	Узел устойчивый	Элементарная матрица *а*
2	Узел неустойчивый	Элементарная матрица *б*
3	Звездный узел устойчивый	Матрица отражения
4	Звездный узел неустойчивый	Матрица вращения
5	Вырожденный узел устойчивый	Элементарная матрица *а*
6	Вырожденный узел неустойчивый	Элементарная матрица *б*
7	Седло	Матрица отражения
8	Центр	Матрица вращения
9	Устойчивый фокус	Элементарная матрица *а*
10	Неустойчивый фокус	Элементарная матрица *б*
11	Узел устойчивый	Матрица отражения
12	Узел устойчивый	Матрица вращения

В качестве матриц $\mathbf{s}$ использовать:

а) элементарную матрицу

$$S = E + u \cdot v^T; \quad S^{-1} = E - u \cdot v^T, \quad (u, v) = 0;$$

б) элементарную матрицу

$$S = E + u \cdot v^T; \quad S^{-1} = E - \frac{1}{1+d} u \cdot v^T, \quad (u,v) = d;$$

в) матрицу отражения

$$H = E - \alpha \cdot u \cdot u^T; \quad \alpha = \frac{2}{u^T \cdot u}; \quad H^{-1} = H;$$

г) матрицу плоских вращений Гивенса

$$G(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}; \quad G^{-1}(\alpha) = G(-\alpha).$$

Собственные числа матрицы выбирать так, чтобы особая точка оказалась такой, как указано в табл. 1.4.

ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Рассмотрим модель «физический маятник» (рис. 1.16). Уравнение модели (рис. 1.17) — это уравнение второго порядка, которое может быть приведено к системе двух уравнений первого порядка. Таким образом, мы можем говорить о построении фазового портрета системы на плоскости. «Красивый» фазовый портрет подразумевает построение нескольких траекторий, начинающихся в различных начальных точках плоскости.

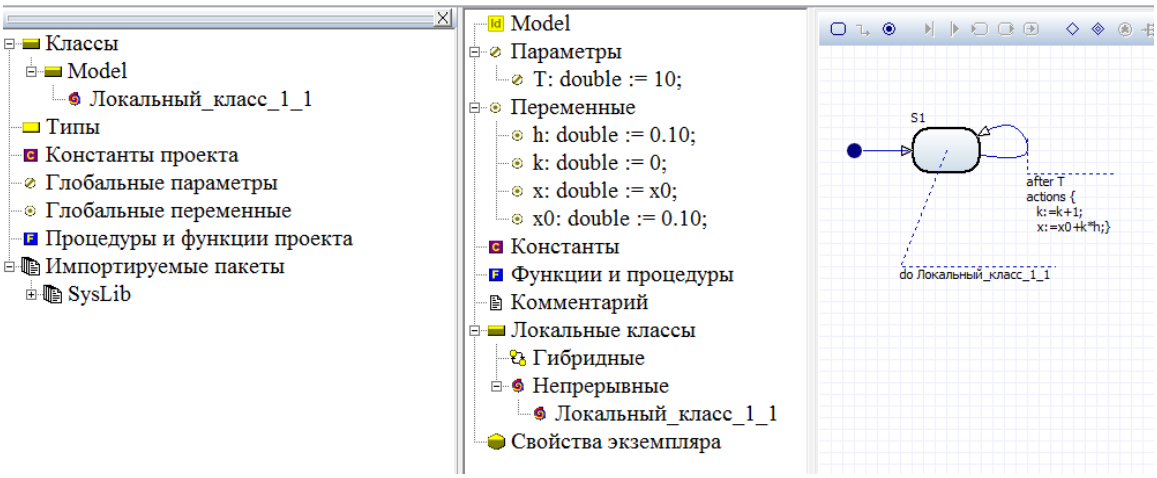


Рис. 1.16. «Физический маятник» в среде RMD

На рис. 1.16 показан гибридный автомат, используемый для рисования фазовых портретов. Действительно, если состоянию S1 при-

писано поведение, соответствующее «физическому маятнику (рис. 1.17), то через каждые T секунд модельного времени будет рисоваться новая траектория, соответствующая новым начальным условиям.

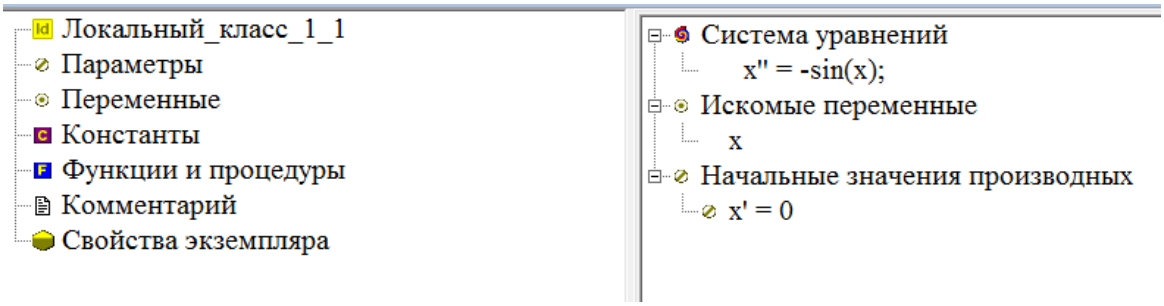


Рис. 1.17. Уравнения «физического маятника»
в виде локального класса

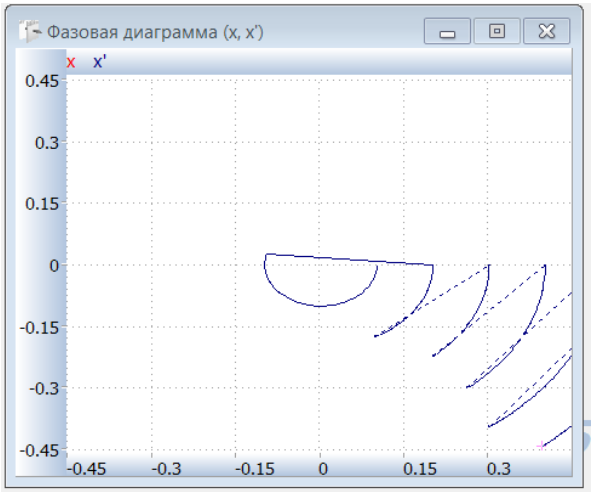


Рис. 1.18. Странный фазовый портрет

Мы ожидаем увидеть замкнутые траектории, а видим нечто странное (рис. 1.18). Обратите внимание, в окне настроек отключен режим «автомасштабирование». Может быть неправильно выбраны настройки окна «Параметры диаграммы» (рис. 1.19)? Нет, настройки выбраны правильно, и режим «автомасштабирование» в этом случае действительно должен быть отключен, так как мы сами выбираем масштабы по осям. Просто мы неправильно выбрали значение параметра T и не успеваем за этот промежуток «замкнуть» траекторию. Выберите правильное значение параметра, и вы увидите желаемые замкнутые траектории.

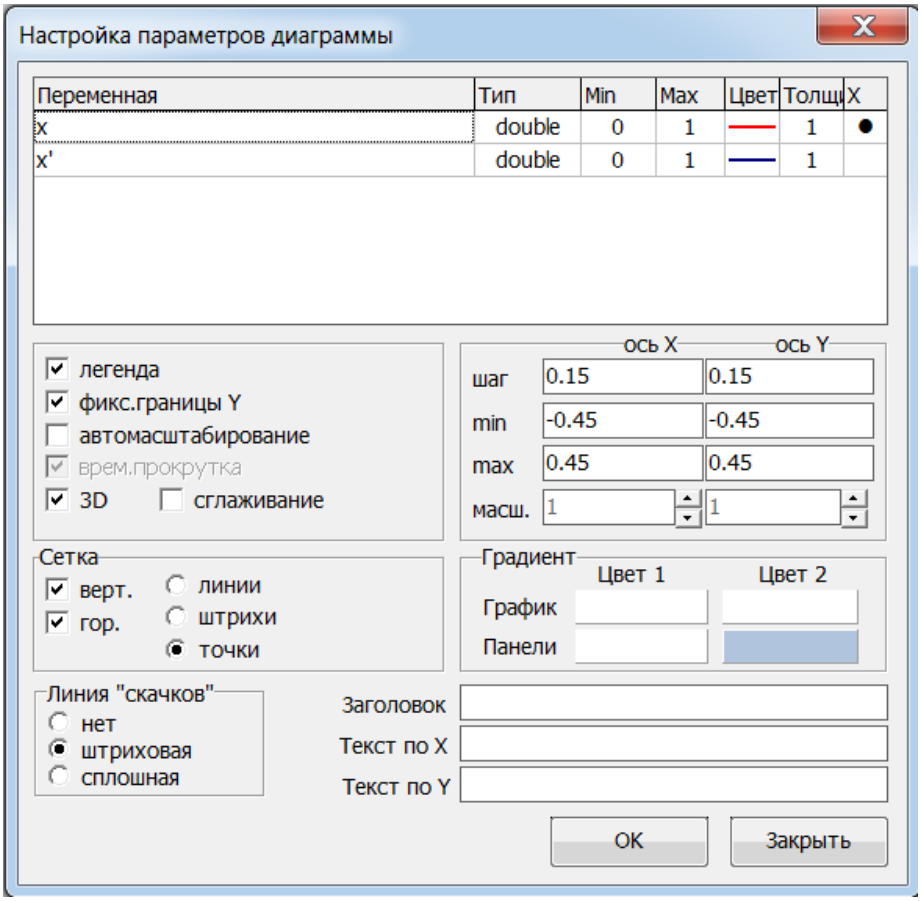


Рис. 1.19. Настройки окна «Параметры диаграммы»

ЗАДАНИЕ 5. ЛИНЕАРИЗАЦИЯ

Найти неподвижные точки, линеаризовать систему в их окрестности и построить фазовые портреты исходной и линеаризованной систем в окрестности особой точки в среде Matlab, Maple.

1. $\begin{cases} \dfrac{dx_1}{dt} = -\dfrac{1}{2}(x_1 + x_2) + x_2^{\,2} \\ \dfrac{dx_2}{dt} = \dfrac{1}{2}(-x_1 + 3x_2) - x_1^{\,2} \\ \dfrac{dx_3}{dt} = -\dfrac{1}{2}x_3 \end{cases}$.

2. $\begin{cases} \dfrac{dx_1}{dt} = x_1(x_1 + 2x_2 + 3) \\ \dfrac{dx_2}{dt} = x_2(2x_1 + x_2 + 3) \\ \dfrac{dx_3}{dt} = \dfrac{1}{2}x_3 \end{cases}$.

3. $\begin{cases} \dfrac{dx_1}{dt} = x_1(x_1 - x_2) \\ \dfrac{dx_2}{dt} = x_2(2x_1 - x_2 - 1). \\ \dfrac{dx_3}{dt} = -\dfrac{1}{2}x_3 \end{cases}$

4. $\begin{cases} \dfrac{dx_1}{dt} = x_2 - {x_1}^5 \\ \dfrac{dx_2}{dt} = -x_1 + {x_2}^2. \\ \dfrac{dx_3}{dt} = x_3 \end{cases}$

5. $\begin{cases} \dfrac{dx_1}{dt} = 2x_1 - 2{x_2}^2 \\ \dfrac{dx_2}{dt} = -x_2 + x_1x_2. \\ \dfrac{dx_3}{dt} = \dfrac{1}{2}x_3 \end{cases}$

6. $\begin{cases} \dfrac{dx_1}{dt} = x_1 + x_1{x_2}^3(1 - {x_1}^2) - 1 \\ \dfrac{dx_2}{dt} = 2x_1 - 3x_2 \\ \dfrac{dx_3}{dt} = \dfrac{1}{2}x_3 \end{cases}$.

7. $\begin{cases} \dfrac{dx_1}{dt} = (-{x_1}^2 + \sin x_2 + x_1 - 1)x_2 \\ \dfrac{dx_2}{dt} = (-{x_2}^2 + \cos x_2 + \dfrac{\pi^2}{4})x_1 \\ \dfrac{dx_3}{dt} = -x_3 \end{cases}$.

8. $\begin{cases} \dfrac{dx_1}{dt} = ({x_1}^2 - e^{x_2})x_1 \\ \dfrac{dx_2}{dt} = -x_2(1 + x_1) \\ \dfrac{dx_3}{dt} = -\dfrac{1}{2}x_3 \end{cases}$.

9. $\begin{cases} \dfrac{dx_1}{dt} = (3x_1 + {x_2}^2 - 4)x_1 \\ \dfrac{dx_2}{dt} = {x_1}^2 - {x_2}^2 \\ \dfrac{dx_3}{dt} = -\dfrac{1}{2}x_3 \end{cases}$.

10. $\begin{cases} \dfrac{dx_1}{dt} = x_1 - x_2 - {x_2}^3 \\ \dfrac{dx_2}{dt} = -x_1 + x_2 - {x_1}^3. \\ \dfrac{dx_3}{dt} = -\dfrac{1}{2}x_3 \end{cases}$

11. $\begin{cases} \dfrac{dx_1}{dt} = \sin(x_1 + x_2) \\ \dfrac{dx_2}{dt} = (x_1 - \pi)x_2 \\ \dfrac{dx_3}{dt} = -\dfrac{1}{2}x_3 \end{cases}$.

12. $\begin{cases} \dfrac{dx_1}{dt} = (x_1 - x_2 + e^{x_1} - 1)x_1 \\ \dfrac{dx_2}{dt} = (x_1 - x_2 - 1)x_2 \\ \dfrac{dx_3}{dt} = \dfrac{1}{2}x_3 \end{cases}$.

13. $\begin{cases} \dfrac{dx_1}{dt} = x_2 x_1 (x_1 - 1)(x_1 + 1) \\ \dfrac{dx_2}{dt} = (2 - x_1^2 - {x_1}^4)x_2 \\ \dfrac{dx_3}{dt} = \dfrac{1}{2}x_3 \end{cases}$.

14. $\begin{cases} \dfrac{dx_1}{dt} = (x_1 x_2 - x_2)(x_1 x_2 + x_2) \\ \dfrac{dx_2}{dt} = {x_2}^2 - {x_1}^2 \\ \dfrac{dx_3}{dt} = \dfrac{1}{2}x_3 \end{cases}$.

ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Вновь обратимся к модели «физический маятник». Одновременно с ней рассмотрим «математический маятник» (рис. 1.20). Очевидно, что «математический маятник» является линеаризованной моделью «физического маятника»: в малой окрестности особой точки x = 0 функцию sin(x) можно заменить первым членом разложения в ряд Тейлора.

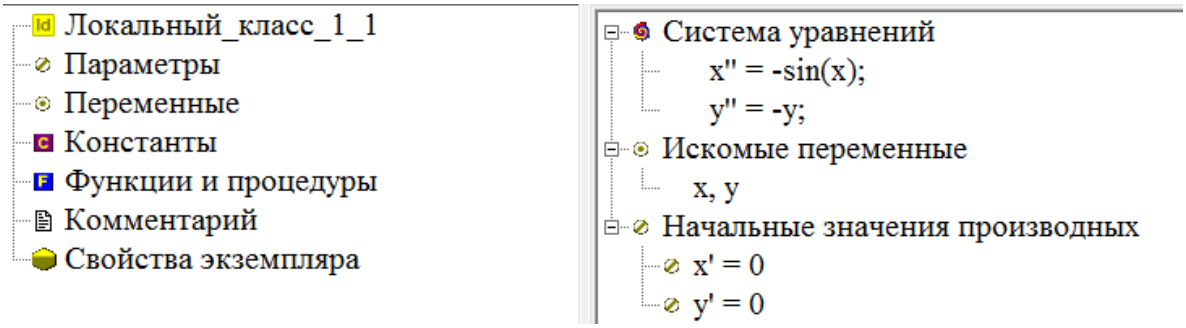


Рис. 1.20. Уравнения «физического» и «математического» маятника

Можно ли в этом случае утверждать, что нулевое положение равновесия устойчиво по первому приближению?

На рис. 1.21 хорошо видно, что фазовые портреты исходной и линеаризованной систем совпадают.

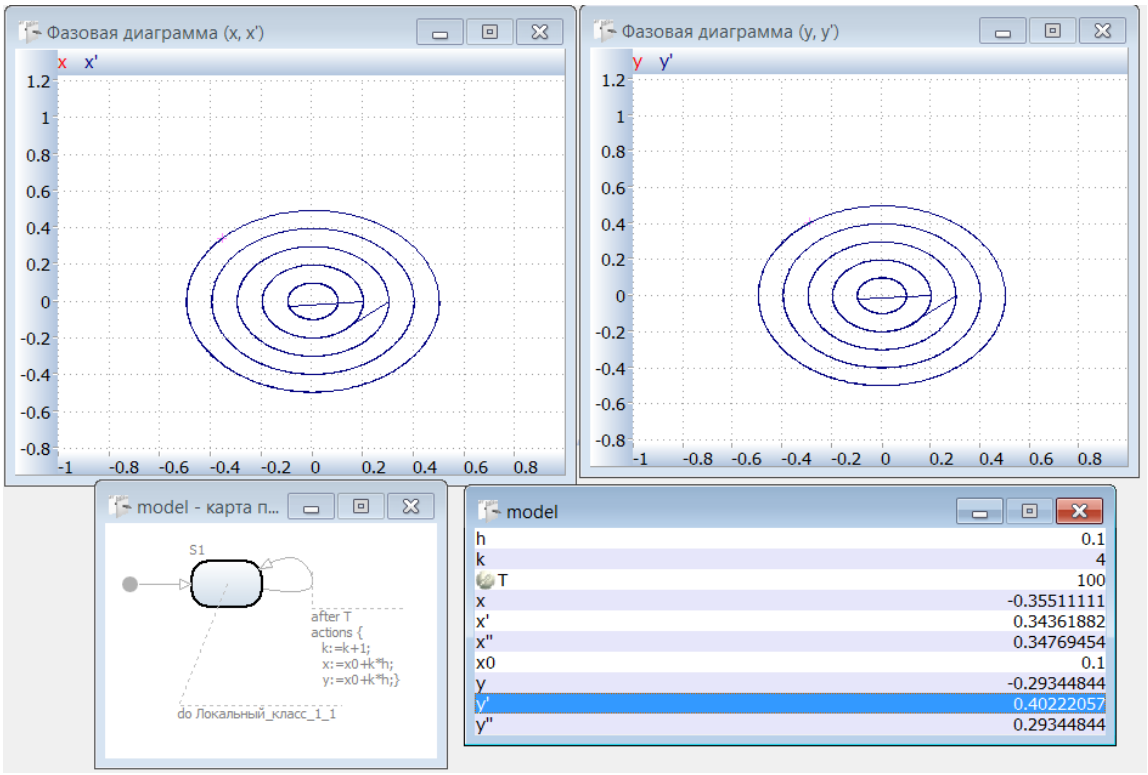


Рис. 1.21. Фазовые портреты «физического» и «математического» маятников

Окно «Гибридный автомат» (в левом нижнем углу на рис. 1.21) показывает, как строятся отдельные траектории.

ЗАДАНИЕ 6. ЧИСЛЕННОЕ И СИМВОЛЬНОЕ РЕШЕНИЕ

Задачи 6.1 – 6.2.

Для заданной системы уравнений

$$\frac{dx}{dt} = A \cdot x + b; \quad x_0 = \begin{bmatrix} 10 \\ -10 \end{bmatrix}; \quad b = \begin{bmatrix} -10 \\ 10 \end{bmatrix};$$

$$A = \alpha E + z \cdot z^T; \; a = -10; \quad E = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \; z = \begin{bmatrix} 1 \\ 2 \end{bmatrix};$$

$$t \in [1,\ 10]$$

найти:

1. Численное решение, используя Matlab.

2. Символьное решение, используя Maple.

А. Реализовать явный метод Эйлера

$$y_{n+1} = y_n + h(Ay_n + b).$$

Б. Реализовать неявный метод Эйлера

$$y_{n+1} = y_n + h(Ay_{n+1} + b).$$

Используя символьное решение $x(t_n)$, найденное в Maple, и численное, найденное с помощью Matlab, построить график ошибки

$$error = \frac{\|x(t_n) - y_n\|}{\|x(t_n)\|}.$$

Задачи 6.3 – 6.4.

Для заданной системы уравнений

$$\frac{dx}{dt} = A \cdot x + b; \quad x_0 = \begin{bmatrix} -10 \\ 10 \end{bmatrix}; \quad b = \begin{bmatrix} 10 \\ -10 \end{bmatrix};$$

$$A = \alpha E + z \cdot z^T; \ \alpha = -5; \quad E = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \ z = \begin{bmatrix} 2 \\ 1 \end{bmatrix};$$

$$t \in [1,\ 10]$$

найти:

1. Численное решение, используя Matlab.

2. Символьное решение, используя Maple.

А. Реализовать явный метод Эйлера

$$y_{n+1} = y_n + h(Ay_n + b).$$

Б. Реализовать неявный метод Эйлера

$$y_{n+1} = y_n + h(Ay_{n+1} + b).$$

Можно ли утверждать, что при $n \to \infty$ и $x(t_\infty) = A^{-1}b$ $\text{error} \to 0$,

$$error = \frac{\|x(t_\infty) - y_n\|}{\|x(t_\infty)\|}.$$

Задачи 6.5 – 6.6

Для заданной системы уравнений

$$\frac{dx}{dt} = A \cdot x + b; \quad x_0 = \begin{bmatrix} 10 \\ -10 \end{bmatrix}; \quad b = \begin{bmatrix} -10 \\ 10 \end{bmatrix};$$

$$A = \begin{bmatrix} \mu & \nu \\ -\nu & \mu \end{bmatrix}; \ \mu = -1; \nu = 2;$$

$$t \in [1,\ 10]$$

найти:

1. Численное решение, используя Matlab.
2. Символьное решение, используя Maple.

А. Реализовать явный метод Эйлера

$$y_{n+1} = y_n + h(Ay_n + b).$$

Б. Реализовать неявный метод Эйлера

$$y_{n+1} = y_n + h(Ay_{n+1} + b).$$

Используя символьное решение $x(t_n)$, найденное в Maple, и численное решение, найденное с помощью Matlab, построить график ошибки

$$error = \frac{\|x(t_n) - y_n\|}{\|x(t_n)\|}.$$

Задача 6.7.

Для заданной системы уравнений

$$\frac{dx}{dt} = A \cdot x + b; \quad x_0 = \begin{bmatrix} 10 \\ -10 \end{bmatrix}; \quad b = \begin{bmatrix} -10 \\ 10 \end{bmatrix};$$

$$A = \alpha E + z \cdot z^T; \ a = -10; \quad E = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \ z = \begin{bmatrix} 1 \\ 2 \end{bmatrix}; \qquad (1.3)$$

$$t \in [1,\ 10]$$

найти:

1. Численное решение, используя Matlab. Показать, что при $t \to \infty$ решение $x(t) \to -A^{-1}b$.

2. Такие α, β, γ для уравнения

$$\frac{d^2z}{dt^2} + \alpha\frac{dz}{dt} + \beta z + \gamma = 0; \left.\frac{dz}{dt}\right|_{t=0} = ?; z(0) = ? \qquad (1.4)$$

чтобы его решение совпало с решением системы (1.3). Решить (1.4), используя Maple, и построить фазовый портрет.

Задача 6.8.

Для заданной системы уравнений

$$\frac{dx}{dt} = A \cdot x + b; \quad x_0 = \begin{bmatrix} 10 \\ -10 \end{bmatrix}; \quad b = \begin{bmatrix} -10 \\ 10 \end{bmatrix};$$

$$A = \begin{bmatrix} \mu & \nu \\ -\nu & \mu \end{bmatrix}; \ \mu = -1; \nu = 2; \qquad (1.5)$$

$$t \in [1,\ 10]$$

найти:

1. Численное решение, используя Matlab.

2. Такие α, β, γ для уравнения

$$\frac{d^2z}{dt^2} + \alpha\frac{dz}{dt} + \beta z + \gamma = 0; \left.\frac{dz}{dt}\right|_{t=0} = ?; z(0) = ? \qquad (1.6)$$

Чтобы его решение совпало с решением системы (1.5). Решить (1.6), используя Maple, и построить фазовый портрет.

ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Предположим, что нам необходимо решить дифференциальное уравнение

$$\frac{dx}{dt} = Ax + b; x(0) = x_0; t \in [0, T]$$

с квадратной постоянной матрицей второго порядка, найти его решения в символьной форме, численно и сравнить оба решения по точности.

При численных вычислениях мы должны задать все значения коэффициентов матрицы *A* и вектора *b*, а также конкретные начальные условия и длину промежутка.

Пакет Maple позволяет написать решение в символьном виде, то есть в форме

$$x(t) = e^{At}x_0 + A^{-1}(e^{At} - E)b,$$

при этом можно получить как приведенную векторно-матричную форму, так и запись решения в координатной форме. Сейчас мы будем говорить о решении в координатной форме.

И та, и другая форма предполагают вычисление собственных чисел матрицы A, что в общем случае невозможно сделать точно, однако в наших примерах размерность системы не будет превышать четырех, что позволяет нам рассчитывать на получение решения в символьном виде.

Для работы нам понадобятся два пакета — linag и DETools, первый позволяет работать с матрицами, а второй позволяет решать дифференциальные уравнения. Начнем с определения необходимых нам математических объектов.

```
> restart;with(linalg);A:=matrix(2,2,[[a,b],[c,d]]);
```

```
Warning, the protected names norm and trace
have been redefined and unprotected
```

```
[BlockDiagonal, GramSchmidt, JordanBlock, LUdecomp, QRdecomp, Wronskian,
  addcol, addrow, adj, adjoint, angle, augment, backsub, band, basis, bezout,
  blockmatrix, charmat, charpoly, cholesky, col, coldim, colspace, colspan,
  companion, concat, cond, copyinto, crossprod, curl, definite, delcols, delrows, det,
  diag, diverge, dotprod, eigenvals, eigenvalues, eigenvectors, eigenvects, entermatrix,
  equal, exponential, extend, ffgausselim, fibonacci, forwardsub, frobenius,
  gausselim, gaussjord, geneqns, genmatrix, grad, hadamard, hermite, hessian,
  hilbert, htranspose, ihermite, indexfunc, innerprod, intbasis, inverse, ismith,
  issimilar, iszero, jacobian, jordan, kernel, laplacian, leastsqrs, linsolve, matadd,
  matrix, minor, minpoly, mulcol, mulrow, multiply, norm, normalize, nullspace,
  orthog, permanent, pivot, potential, randmatrix, randvector, rank, ratform, row,
  rowdim, rowspace, rowspan, rref, scalarmul, singularvals, smith, stackmatrix,
  submatrix, subvector, sumbasis, swapcol, swaprow, sylvester, toeplitz, trace,
  transpose, vandermonde, vecpotent, vectdim, vector, wronskian]
```

$$A := \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

В приведенном фрагменте стоит прокомментировать некоторые особенности: при вызове пакета перечисляются все содержащиеся в нем процедуры; пакет различает большие и маленькие буквы; начиная работать с новой задачей, стоит приготовиться к тому, что вы не раз ошибетесь и будете многократно повторять вычисления — команда «restart» позаботится о стирании ненужной информации.

Определим вектор правой части и дадим имена $x(t), y(t)$ компонентам искомого решения X. Вектора Y будет обозначать правую часть нашей системы.

```
> B:=vector([n,m]);X:=vector(2,[x(t),y(t)]);
```

$$B := [n, m]$$

$$X := [\mathrm{x}(t), \mathrm{y}(t)]$$

```
> Y:=vector(2);
```

$$Y := \mathrm{array}(1 .. 2, [\ ])$$

Сложность и скорость символьных вычислений во многом зависит от того, какие символьные объекты участвуют в вычисления, и сколько их. Для того чтобы удобнее было проверять себя, выберем матрицу диагональной, оставив пока правую часть произвольной. Вектор *Y* хранит правую часть в обще виде, а *Y1* — с учетом сделанной подстановки.

```
> Y:=matadd(multiply(A,X),B);Y1:=subs(a=-
1,b=0,c=0,d=-2,matadd(multiply(A,X),B));
```

$$Y := [a\,\mathrm{x}(t) + b\,\mathrm{y}(t) + n,\ c\,\mathrm{x}(t) + d\,\mathrm{y}(t) + m]$$

$$Y1 := [-\mathrm{x}(t) + n,\ -2\,\mathrm{y}(t) + m]$$

Для формирования решения нам понадобиться процедура dsolve, которая умеет решать системы уравнений первого порядка. Вызвав соответствующий раздел справки, вы увидите, как необходимо формировать саму систему и начальные условия, и поймете, почему мы дали компонентам решения имена $x(t), y(t)$.

>

```
diff_eq:=Diff(x(t),t)=Y1[1],Diff(y(t),t)=Y1[2];
```

$$diff\_eq := \frac{\partial}{\partial t} x(t) = -x(t) + n, \frac{\partial}{\partial t} y(t) = -2\, y(t) + m$$

```
> init_con:=x(0)=-1,y(0)=1;
```

$$init\_con := x(0) = -1, y(0) = 1$$

```
> solu-
tion:=dsolve({diff_eq,init_con},{x(t),y(t)});
```

$$solution := \{ y(t) = \frac{1}{2} m + \mathbf{e}^{(-2t)} \left(-\frac{1}{2} m + 1 \right), x(t) = n + \mathbf{e}^{(-t)} (-n - 1) \}$$

Полученное решение легко проверить. Попробуйте получить решение в общем виде, не указывая конкретные значения коэффициентов матрицы, и вы поймете, почему мы отказались от записи решения в общем виде. Глядя на полученное решение, мы видим, что оно представлено в виде списка, каждый элемент которого представлен в виде выражения для соответствующей компоненты решения. В эти выражения можно подставлять конкретные значения параметров и получать решения в нужной нам точке.

```
> solution1:=subs(t=10,m=1,n=1,solution);
```

$$solution1 := \{ y(10) = \frac{1}{2} + \frac{1}{2} \mathbf{e}^{(-20)}, x(10) = 1 - 2\, \mathbf{e}^{(-10)} \}$$

Покажем, как можно выделить из выражения отдельные его части, в данном случае правую часть выражения первой компоненты решения.

```
> solution[1];rhs(solution[1]);
```

$$y(t) = \frac{1}{2} m + \mathbf{e}^{(-2t)} \left(-\frac{1}{2} m + 1 \right)$$

$$\frac{1}{2} m + \mathbf{e}^{(-2t)} \left(-\frac{1}{2} m + 1 \right)$$

Подготовимся к дальнейшему, выделим обе правые части символьного решения.

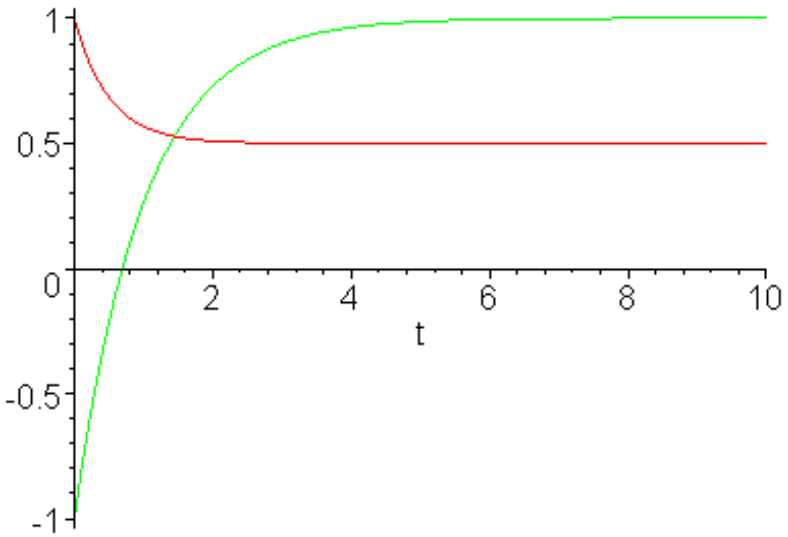
```
expr1:=subs(m=1,rhs(solution[1]));expr2:=subs(
n=1,rhs(solution[2]));
```

$$expr1 := \frac{1}{2} + \frac{1}{2}\,\mathbf{e}^{(-2\,t)}$$

$$expr2 := 1 - 2\,\mathbf{e}^{(-t)}$$

и нарисуем графики:

```
> plot({expr1,expr2},t=0..10);
```



Теперь перейдем к численному решению системы.

```
diff_eq1:=subs(n=1,m=1,Diff(x(t),t)=Y1[1]),sub
s(n=1,m=1,Diff(y(t),t)=Y1[2]);solution_N:=dsolve({
diff_eq1,init_con},numeric);
```

$$diff\_eq1 := \frac{\partial}{\partial t}\,x(t) = -x(t) + 1,\ \frac{\partial}{\partial t}\,y(t) = -2\,y(t) + 1$$

$$solution\_N := \mathbf{proc}(rkf45\_x)\ \ldots\ \mathbf{end\ proc}$$

Имя solution_N в данном случае — это имя процедуры от одного параметра (времени), которая позволяет нам найти численное решение в произвольной точке. Напишем фрагмент, который одновременно формирует символьное и численное решение на заданном множестве точек.

```
> t:=0;
h:=0.25;for i from 1 by 1 while i < 6 do
```

```
    TbE[i]:=evalf(t),evalf(expr2),evalf(expr1);
TbN[i]:=solution_N(t);  t := t + h
    end do;
```

и посмотрим, что мы получим.

$$t := 0$$

$$h := .25$$

$$TbE_1 := 0., -1., 1.$$

$$TbN_1 := [0 = 0., x(t) = -1., y(t) = 1.]$$

$$t := .25$$

$$TbE_2 := .25, -.557601566, .8032653298$$

$$TbN_2 := [.25 = .25, x(t) = -.557601559868365904, y(t) = .803265244470526607]$$

$$t := .50$$

$$TbE_3 := .50, -.213061319, .6839397206$$

$$TbN_3 := [.50 = .50, x(t) = -.213061309944886507, y(t) = .683939620155625038]$$

$$t := .75$$

$$TbE_4 := .75, .0552668946, .6115650800$$

$$TbN_4 := [.75 = .75, x(t) = .0552669063804118871, y(t) = .611564982069755625]$$

$$t := 1.00$$

$$TbE_5 := 1.00, .2642411176, .5676676416$$

$$TbN_5 := [1.00 = 1.00, x(t) = .264241132321481498, y(t) = .567667546855770810]$$

$$t := 1.25$$

Мы видим, что оба массива содержат нужную нам информацию, но в различной форме. Осталось только правильно найти нужные данные и вычислить итоговую таблицу.

```
> for i from 1 by 1 while i < 6 do
    res[i]:=[TbE[i][1],abs((TbE[i][2]-
rhs(TbN[i][2]))/TbE[i][2]),abs((TbE[i][3]-
rhs(TbN[i][3]))/TbE[i][3])]end do;
```

$$res_1 := [0., 0., 0.]$$

$$res_2 := [.25, .109397110\,3\,10^{-7}, .106191561\,9\,10^{-6}]$$

$$res_3 := [.50,\ .427107090210^{-7},\ .146796562610^{-6}]$$

$$res_4 := [.75,\ .213147492510^{-6},\ .160081082510^{-6}]$$

$$res_5 := [1.00,\ .556310090310^{-7},\ .166822966610^{-6}]$$

2. ОДНОКОМПОНЕНТНЫЕ ГИБРИДНЫЕ СИСТЕМЫ

Задания второго раздела предназначены для изучения более сложных моделей — гибридных систем.

«Задание 1» — это лабораторная работа по дисциплине «Моделирование», раздел «Моделирование динамических систем». Эта работа выполняется в среде Rand Model Designer. Основная цель — познакомить студента с понятием «гибридная система», более сложным типом модели, чем «динамическая система», и научить строить такие системы, используя графические конструкции языков моделирования «машина состояний» или «гибридный автомат».

Выдавая задание, следует пояснить, что в данном случае студент имеет дело с кусочно-непрерывной функцией, и ее нужно реализовать, используя конструкцию «гибридный автомат». Каждое состояние автомата соответствует классической динамической системе.

Правая часть уравнения, или кусочно-непрерывная функция, представленная гибридным автоматом, содержит свободные коэффициенты, позволяющие «управлять» поведением системы. Выбирая значения части коэффициентов нулевыми можно получить: 1) модель математического маятника; 2) модель математического маятника, подверженного гармоническому возмущению, приводящему к резонансу; 3) кусочно-линейную модель, приводящую к периодическим колебаниям сложной формы.

Формализмы «машина состояний» и «гибридный автомат» достаточно сложны для понимания, поэтому в этом разделе сначала приведены более простые примеры гибридных систем (Задание 1), а затем они усложняются (Задание 2). Гибридные системы, как в свое время и «колебательные» системы, выделились в отдельные дисци-

плины «Теория гибридных систем» и «Теория колебаний». Примеров гибридных систем очень много в механике. Отобранные в качестве примеров механические системы наглядны, просты с точки зрения уравнений, определяющих их поведение. Еще одно их очень важное достоинство — их поведение легко представить с помощью двумерной или трехмерной анимации, доступной в современных пакетах моделирования. Таким образом, эти задания можно использовать для изучения средств визуализации, предоставляемых пакетом моделирования.

ЗАДАНИЕ 1. «ПРОСТЫЕ» ГИБРИДНЫЕ СИСТЕМЫ

Части задач 1 – 6 заимствованы из книги [2].

1. ДВИЖЕНИЕ ЧАСТИЦЫ

Частица начинает движение в плоскости (x,y) с постоянной скоростью $V_x = c1$ и $V_y = c2$ из начальной точки (x_0,y_0) и под действием гравитационного поля внутри:

a) квадрата со стороной d;

b) равнобедренного прямоугольного треугольника с катетами длиной d;

c) равностороннего треугольника со стороной равной d.

Соударение со стенками — абсолютно упругое. Что будет происходить с траекториями, если гравитационное поле: а) отсутствует; б) периодически «выключается»; в) присутствует всегда. Управление полем реализовать с помощью гибридного автомата.

2. МАЯТНИК С ОГРАНИЧЕНИЯМИ

Плоские колебания маятника в виде шара массы *m*, подвешенного на невесомом и абсолютно упругом стержне длины L, ограничены:

а) двумя абсолютно упругими вертикальными стенками, расположенными на расстоянии d от точки подвеса;

б) двумя абсолютно упругими упорами, расположенными в точках $(-L_1, L_1)$, (L_1, L_1), $L_1<L$, начало координат в точке подвеса, вертикальная ось «смотрит» вниз;

в) двумя абсолютно упругими упорами, расположенными в точках ($-L_1$, L_1), (L_2, L_2), L_1 , $L_2<L$, начало координат в точке подвеса, вертикальная ось «смотрит» вниз.

Удар о стенки или любой упор — абсолютно упругие.

На маятник также действует синусоидальная возмущающая сила

$$\frac{d^2x}{dt^2} + a^2 * x = k^2 * \sin(t), a > k.$$

Что будет происходить с маятником, если синусоидальное воздействие: а) отсутствует; б) периодически «выключается»; в) присутствует всегда.

3. ГРУЗИК НА ПРУЖИНКЕ

Груз массы *m* прикреплен к правому концу пружины, левый конец которой закреплен в стене (рис. 2.1).

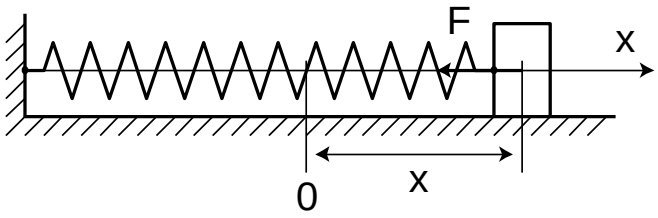


Рис. 2.1. К задаче 3

Ось *x* направлена вдоль оси пружины, начало отсчета находится в правом конце недеформированной пружины.

В некоторый момент времени груз начинает двигаться с начальной скоростью v_0, направленной по горизонтали направо. При достижении своего максимального положения отклонения от нуля его останавливают и удерживают в этом положении. При каком значении *x* это произойдет?

Уравнения:

покой: $m\ddot{x} = 0$;

движение: $\begin{cases} m\ddot{x} = F_x \\ F_x = -\alpha x - \beta x^3; \alpha, \beta > 0 \end{cases}$,

где F_x — проекция силы упругости пружины, *x* — удлинение пружины.

Ответ: $x = \frac{1}{\beta}\sqrt{\beta(\sqrt{\alpha^2 + 2m\beta v_0{}^2} - \alpha)}$, x — искомое максимальное отклонение.

4. ТОРМОЖЕНИЕ ЛОДКИ

Лодка массы m имела скорость v_0 (рис. 2.2).

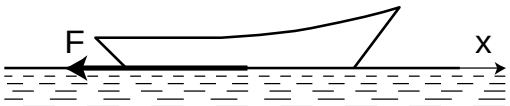


Рис. 2.2. К задаче 4

В некоторый момент времени двигатель выключили, и на нее стала действовать только сила сопротивления $F_x = -\alpha\dot{x} - \beta\dot{x}^2$.

Когда скорость лодки стара в 3 раза меньше начальной, ее остановили. Сколько длилось движение лодки (T) после выключения двигателя, и какой путь (S) она прошла.

Уравнения:

движение: $m\ddot{x} = 0$;

выключен двигатель: $\begin{cases} m\ddot{x} = F_x \\ F_x = -\alpha\dot{x} - \beta\dot{x}^2; \alpha, \beta > 0 \end{cases}$,

где F_x — сила сопротивления воды движению лодки.

Ответ: $S = \frac{m}{\beta}\ln\frac{3(\alpha + \beta v_0)}{3\alpha + \beta v_0}$, $T = \frac{m}{\alpha}\ln\frac{3\alpha + \beta v_0}{\alpha + \beta v_0}$.

5. АВАРИЙНОЕ ПАДЕНИЕ ГРУЗА

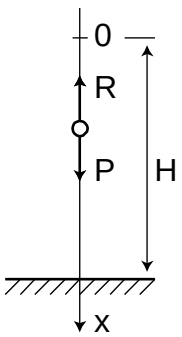


Рис. 2.3. К задаче 5

Груз опускали тросом на дно реки со скоростью v_0 (рис. 2.3). На расстоянии H от дна трос порвался, и груз начал свободно падать. Падение длилось T секунд, найти значение H.

Уравнения:

с тросом: $m\ddot{x} = 0$;

свободное падение: $\begin{cases} m\ddot{x} = P + R_x \\ R_x = -km\dot{x}; k > 0 \\ P = mg \end{cases}$,

где R_x — сила сопротивления воды движению лодки, P — сила тяжести груза.

Ответ: $H = \frac{g}{k}T - \frac{g - kv_0}{k^2}\left(1 - e^{-kT}\right)$.

6. ПРЫЖОК С ПАРАШЮТОМ

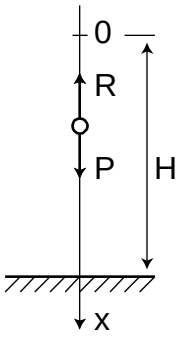


Рис. 2.4. К задаче 6

Парашютист в момент раскрытия парашюта имел скорость v_0, направленную вертикально вниз (рис. 2.4).

Через сколько времени T после раскрытия парашюта он приземлился и с какой скоростью.

Уравнения:

до раскрытия: $m\ddot{x} = -gm$;

после: $\begin{cases} m\ddot{x} = P + R_x \\ R_x = -k^2 m(\dot{x})^2; k > 0 \\ P = mg \end{cases}$,

где R_x — сила сопротивления движению парашютиста, P — сила тяжести.

Уравнение движения парашютиста:

$$x = \frac{1}{k^2}\ln\left(ch(k\sqrt{g}t) + \frac{kv_0}{\sqrt{g}}sh(k\sqrt{g}t)\right),$$

подставляем H для нахождения времени до падения T:

$$H = \frac{1}{k^2}\ln\left(ch(k\sqrt{g}t) + \frac{kv_0}{\sqrt{g}}sh(k\sqrt{g}t)\right).$$

Уравнение для нахождения скорости $v(\dot{x})$:

$$v = \dot{x} = \frac{\sqrt{g}}{k} \frac{\sqrt{g} \cdot th(k\sqrt{g}t) + kv_0}{\sqrt{g} + kv_0 \cdot th(k\sqrt{g}t)}.$$

ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Рассмотрим движение материальной точки в правой полуплоскости ($x \geq 0$) с постоянной скоростью V (составляющие скорости вдоль осей Vx,Vy) до и после удара с вертикальной стенкой, расположенной вдоль оси Y. Горизонтальная составляющая скорости в начальный момент выбрана так, чтобы точка двигалась по направлению к стенке. Удар о стенку упругий. Через 10 единиц модельного времени после столкновения движение заканчивается.

Так как удар о стенку мы считаем «мгновенным», то можно выделить два «длительных» движения (состояния): «движение до удара» и «движение после удара», уравнения, которых имеют вид:

$$\begin{cases} \frac{dx}{dt} = Vx \\ \frac{dy}{dt} = Vy \end{cases}; \qquad \begin{cases} \frac{dx}{dt} = -Vx \\ \frac{dy}{dt} = Vy \end{cases}.$$

На рис. 2.5 показан соответствующий гибридный автомат. Обратите внимание на закомментированную строчку, соответствующую возможным дискретным действиям после наступления события «удар»: (Vx: = –Vx). Этот дискретный оператор позволяет написать только одну систему уравнений, но тогда надо не забыть описать Vx не как параметр, а как переменную. Идя по этому пути, можно сократить и число состояний (форма уравнений не меняется, меняются только значения коэффициентов). Как это сделать?

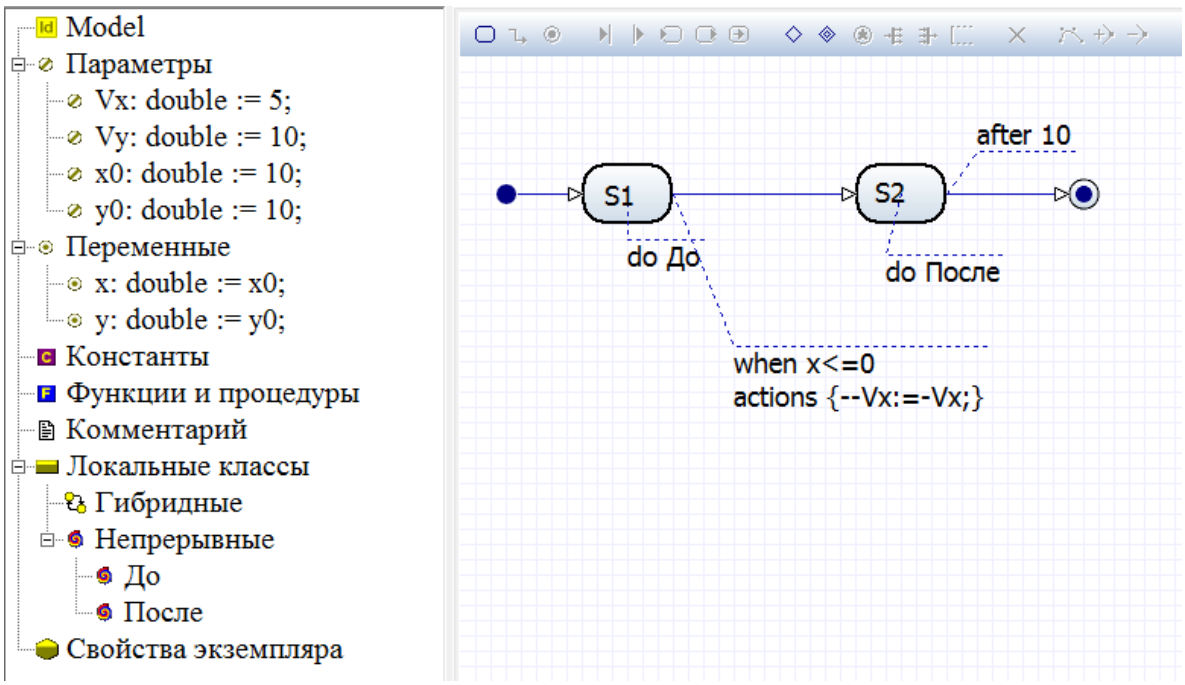


Рис. 2.5. Гибридный автомат, описывающий длительные движения материальной точки (состояния S1, S2) и условие смены состояний

На рис. 2.6 показан график движения точки, точнее, фазовый портрет.

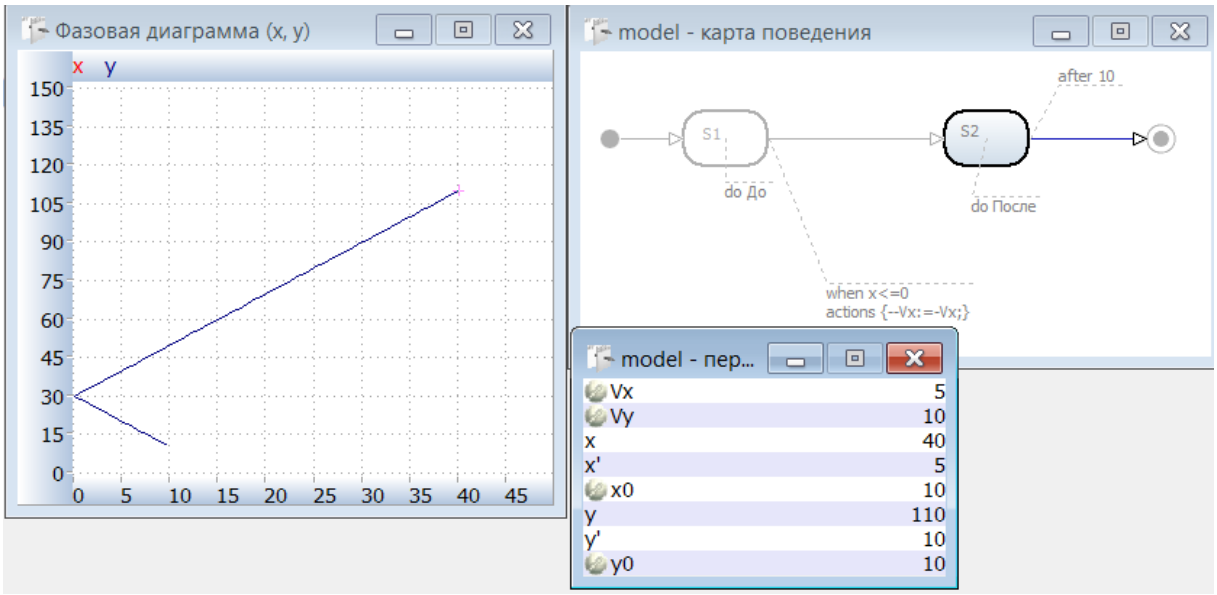
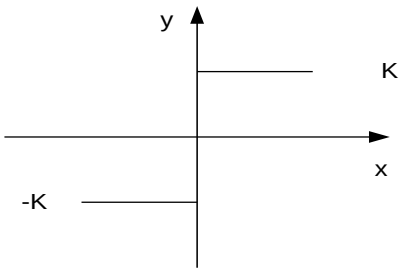
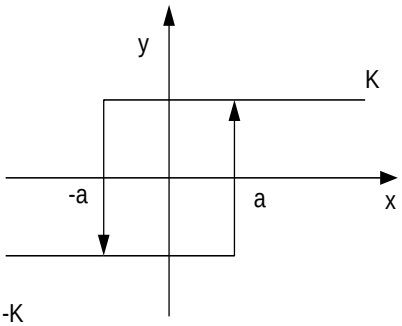


Рис. 2.6. Движение точки до и после удара

ЗАДАНИЕ 2. КУСОЧНО-НЕПРЕРЫВНЫЕ ФУНКЦИИ В ПРАВОЙ ЧАСТИ ДИФФЕРЕНЦИАЛЬНОГО УРАВНЕНИЯ

А. Маятник_1 под воздействие гармонических и кусочно-линейных сил

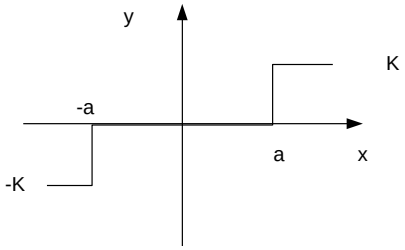


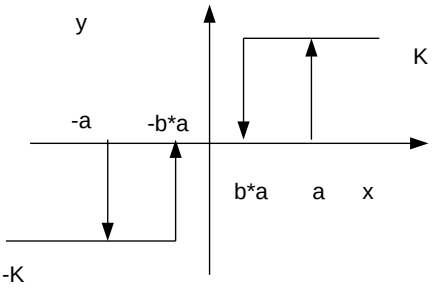


$$y = K \cdot \operatorname{sign}(x)$$

$$y = \begin{cases} K \cdot \operatorname{sign}(x-a), \dfrac{dx}{dt} > 0 \\ K \cdot \operatorname{sign}(x+a), \dfrac{dx}{dt} < 0 \end{cases}$$

а) *б*)





$$y = \frac{K}{2} \cdot \left(\operatorname{sign}(x-a) + \operatorname{sign}(x+a)\right)$$

$$y = \begin{cases} \dfrac{K}{2} \cdot \left(\operatorname{sign}(x-a) + \operatorname{sign}(x+b \cdot a)\right), \dfrac{dx}{dt} > 0 \\ \dfrac{K}{2} \cdot \left(\operatorname{sign}(x+a) + \operatorname{sign}(x-b \cdot a)\right), \dfrac{dx}{dt} < 0 \end{cases},$$

$$b < 1$$

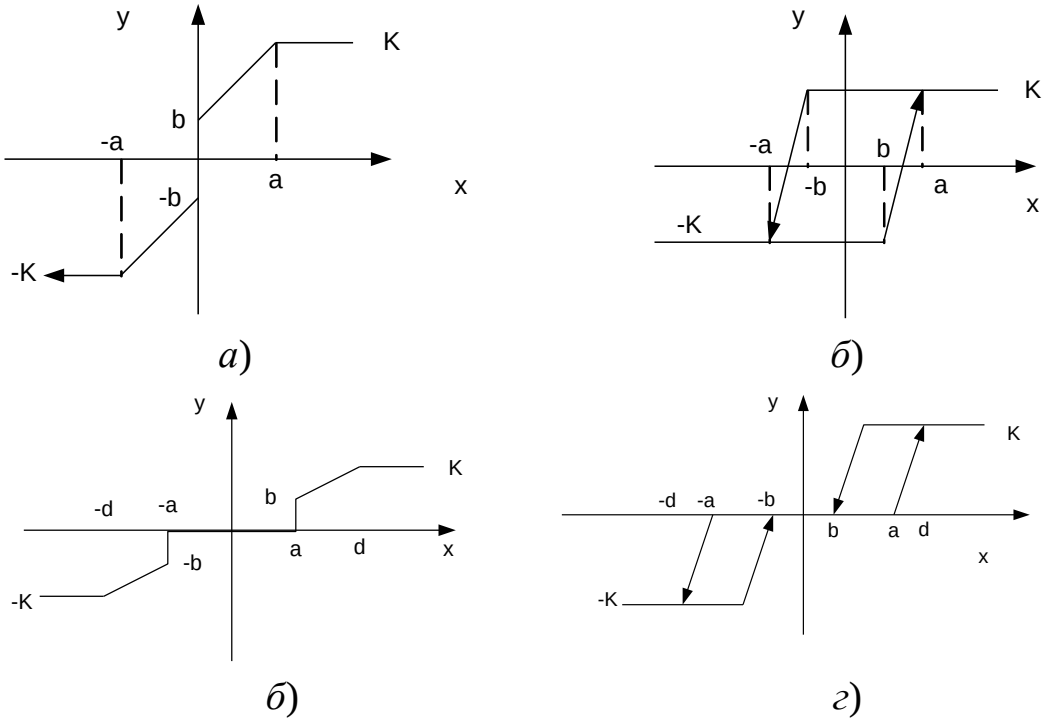
в) *г*)

Рис. 2.7. Нелинейные функции на базе функции sign(x)

Найти численное решение уравнения

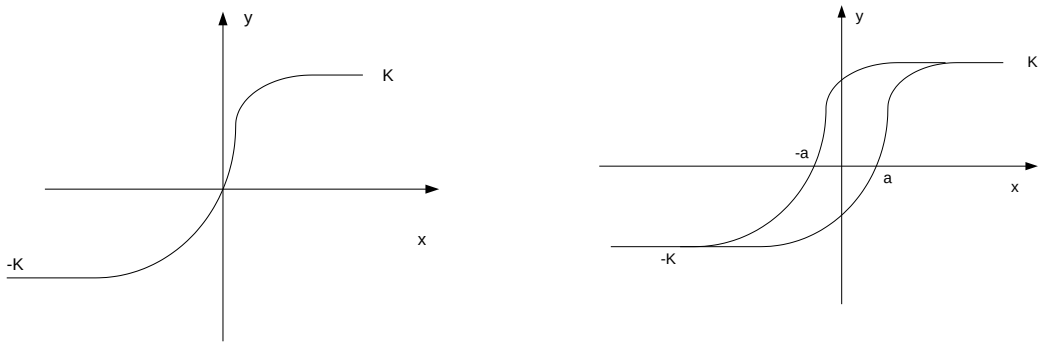
$$\frac{d^2x}{dt^2} = -k_1 F - k_2 \cdot x + k_3 \sin(k_4 \cdot t)$$

с константами k_1, k_2, k_3, k_4 и функцией $F = y$, с различными функциями y (рис. 2.7 – рис. 2.9). Константы, определяющие вид функции (K — максимальное по модулю значение функции, $a > 0, b > 0, d > 0$ — константы, определяющие нули функций), выбрать самостоятельно.



а) *б*) *б*) *г*)

Рис. 2.8. Кусочно-непрерывные функции
на базе линейных функций



$$y = \frac{2 \cdot K}{\pi} arctg(\alpha \cdot x) \qquad y = \frac{K}{\pi} arctg(\alpha \cdot x + a) + \frac{K}{\pi} arctg(\alpha \cdot x - a)$$

а) *б*)

Рис. 2.9. Нелинейные функции на базе функции arctg(x)

Б. Маятник_2 под воздействие гармонических и кусочно-линейных сил

Дано дифференциальное уравнение

$$a \cdot \frac{d^2x}{dt^2} + F(x) = \sin(t); x(0) = x_0,$$

где:

а) $F(x) = k * sign(x), k > 0,$

б) $F(x) = \begin{cases} p * x, abs(x) \le L, p > 0 \\ pL, x > L \\ -pL, x < L \end{cases},$

в) $F(x) = \begin{cases} 0, x \le L, p > 0 \\ p, x > L \\ -p, x > L \end{cases},$

г) $F(x) = \begin{cases} px + c, x > 0, p > 0, c > 0 \\ px - c, x < 0 \end{cases},$

д) $F(x) = \begin{cases} p_1x + c_1, x > L > 0, p_1 > 0, c_1 > 0 \\ p_2x - c_2, x < L, p_2 > 0, c_2 > 0 \end{cases},$

е) $F(x) = \begin{cases} 0, x \le L \\ p_1x + c_1, x > L, p_1L + c_1 = 0 \\ p_1x + c_2, x > L, -p_1L + c_2 = 0 \end{cases}.$

Что будет происходить с траекториями, если синусоидальное воздействие: а) отсутствует; б) периодически «выключается»; в) присутствует всегда.

ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Пример 1. Построим кусочно-непрерывную функцию с помощью гибридного автомата. Рассмотрим функцию sin(t) и ограничим ее значения сверху и снизу числом maximum. Таким образом, получаем кусочно-непрерывную функцию, совпадающую с функций sin(t) вне зоны ограничений и имеющую постоянные значения + maximum и

–maximum при ограничении сверху и снизу (рис. 2.10). Гибридный автомат в этом случае имеет три состояния (рис. 2.11). Поведение автомата определяется тремя непрерывными функциями

$$\begin{cases} x = \sin(Time) \\ y = x \end{cases}; \quad \begin{cases} x = \sin(Time) \\ y = \max imum \end{cases}; \quad \begin{cases} x = \sin(Time) \\ y = -\max imum \end{cases}.$$

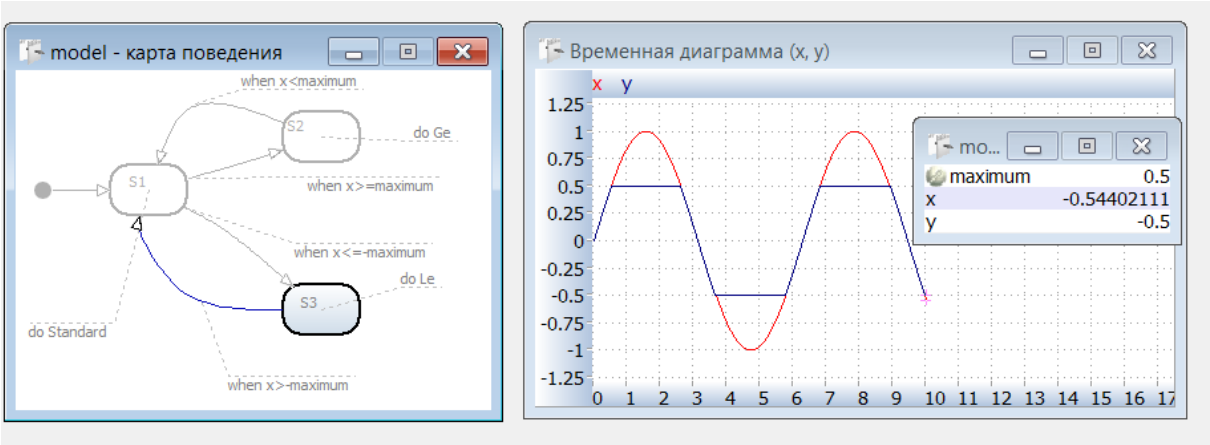


Рис. 2.10. Функция sin(t) и функция sin(t) с ограничениями

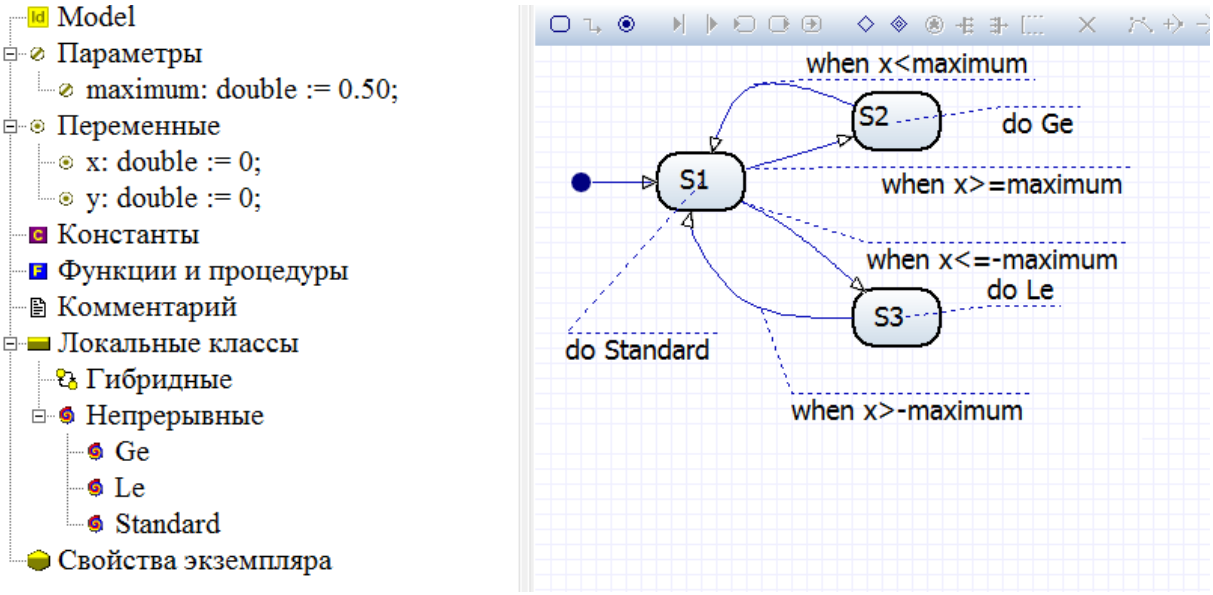


Рис. 2.11. Гибридный автомат кусочно-непрерывной функции с тремя областями определения

Обратите внимание на «системную» переменную Time — это глобальное время модели. Системная переменная «локальное время»

пишется с маленькой буквы — time. Что будет, если использовать локальное время при описании локальных поведений?

Пример 2. Рассмотрим осциллятор

$$m\frac{dx^2}{dt^2} = -k^2 x + S_x; k^2 = \frac{c}{m}, \tag{2.1}$$

где m — масса, c — коэффициент жесткости, S_x — проекция возмущающей силы на ось x. Пусть S_x будет периодической кусочно-постоянной функцией $S_x(t+T) = S_x(t)$ и

$$S_x(t) = \begin{cases} H, & 0 \le t < \dfrac{T}{2} \\ -H, & \dfrac{T}{2} \le t < T \end{cases}. \tag{2.2}$$

Для решения этой задачи можно воспользоваться классическими динамическими системами, аппроксимировав кусочно-постоянную функцию различными непрерывными (аппроксимация правой части уравнения) и гибридной системой.

Рассмотрим первый путь. Используем для аппроксимации ряд Фурье. Разложив периодическую кусочно-постоянную функцию в ряд Фурье

$$S_x = \frac{1}{2}A_0 + \sum_{i=1}^{\infty}(A_i \cos(ipt) + B_i \sin(ipt)), \quad p = \frac{2\pi}{T},$$

$$A_i = \int_0^T S_x(t)\cos(ipt)dt, \quad B_i = \int_0^T S_x(t)\sin(ipt)dt,$$

получим

$$S_x = \frac{4H}{\pi}\sum_{i=1}^{\infty}\frac{1}{i}\sin(ipt), \quad i = 1,3,5,\ldots.$$

Для отрезка ряда с четырьмя членами $i = 1,3,5,7$ при нулевых начальных условиях $x(0) = 0, \dot{x}(0) = 0$ решение уравнения (2.1) дается формулой

$$x(t) = -\frac{hp}{k}(\sum \frac{1}{k^2 - i^2 p^2}) \cdot \sin(k \cdot t)$$

$$+ h\sum \frac{1}{i \cdot (k^2 - i^2 p^2)} \cdot \sin(i \cdot p \cdot t), \qquad (2.3)$$

$$h = \frac{4H}{\pi m}.$$

Первый член в решении (2.3) соответствует свободным колебаниям, второй — вынужденным. Число членов разложения в данном случае не имеет принципиального значения, так как скорость сходимости можно регулировать такими параметрами задачи как масса или коэффициент жесткости.

Второй путь, связанный с гладкими аппроксимациями — это аппроксимация функции $sign(t)$, например функцией $\frac{2}{\pi} arctg(t)$. Для нашей задачи функцию S_x на промежутке $[0, T]$ можно представить в виде

$$S_x = -\frac{2H}{\pi} arctg(\alpha\left(t - \frac{T}{2}\right)), \quad \alpha >> 1. \qquad (2.4)$$

Наконец, можно воспользоваться формулой (2.4) и построить примитивный гибридный автомат, воспроизводящий кусочно-непрерывную функцию (рис. 2.12).

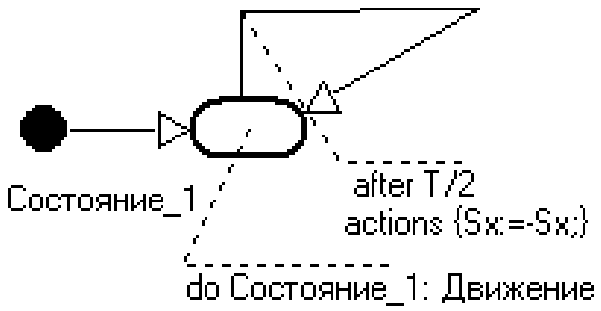


Рис. 2.12. Гибридный автомат
для кусочно-постоянной
функции S_x в форме (2.4)

Результаты аппроксимации и решение показаны на рис. 2.13.

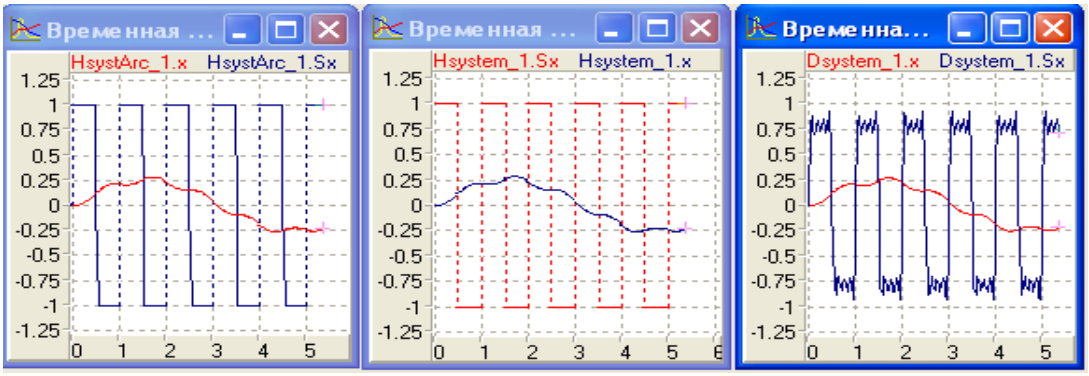


Рис. 2.13. Кусочно-постоянная возмущающая функция (в центре) и ее непрерывные аппроксимации с помощью функции $arctg(t)$ (слева) и отрезка ряда Фурье (справа)

В данном конкретном случае все три численных решения практически одинаковы по точности (если за критерий точности взять их совпадения до заданного числа знаков) и обсуждаться могут лишь вычислительные затраты.

Разложение в ряд Фурье хорошо тем, что позволяет предсказать появление резонанса (рис. 2.14).

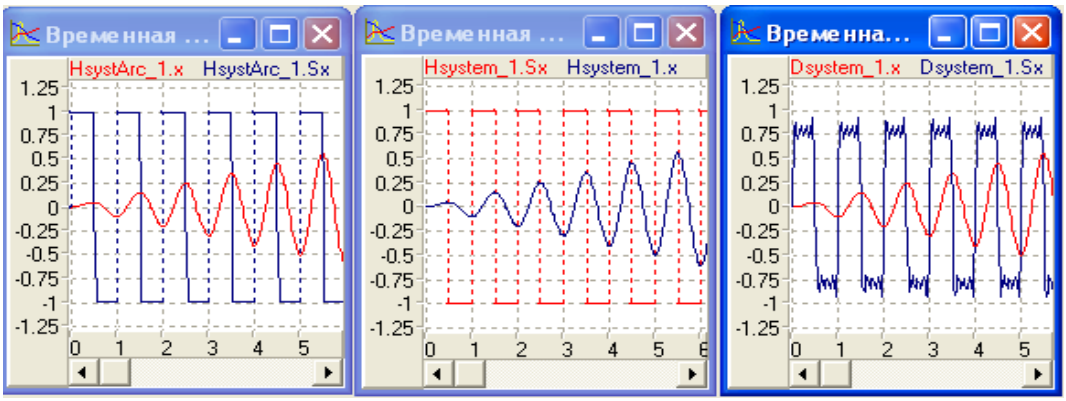


Рис. 2.14. Резонансные колебания первого порядка при различных аппроксимациях правой части

Пример 3. Рассмотрим следующую задачу. Проектируя новый цирк, архитекторы попросили описать все аттракционы, которые входят в программу выступлений. Один из аттракционов оказался таким. Акробат поднимается по вертикальной лестнице до вершины «ка-

тальной горки» длины L — узкой доски, составляющей угол α с ареной и начинает по ней спуск на лыжах. В плоскости (X,Y) горка выглядит так, как показано на рис. 2.15. Спуск по доске переходит в движение по части окружности радиуса r. Скольжение по части окружности (на рис. 2.15 эта часть принадлежит сектору с углом 2α), переходит в свободный полет на поверхностью бассейна, после чего акробат падает в воду и погружается на дно. Нас интересует длина бассейна, обеспечивающая безопасность аттракциона, и нагрузка на поверхности скольжения, если масса акробата m.

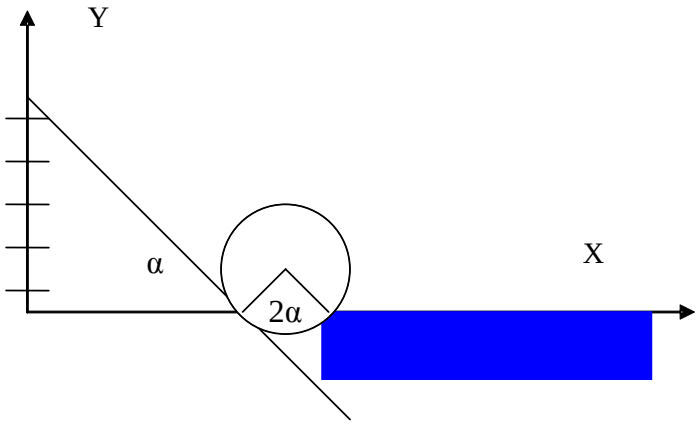


Рис. 2.15. Плоское изображение циркового аттракциона

Эта задача, даже с возможными упрощениями, достаточно сложна, так как предполагает решение нескольких последовательных задач — скольжение по наклонной плоскости, движение по кривой, свободный полет и погружение в воду. Мы построим и модель подъема, но это часть аттракциона не связана с решением нашей задачи о безопасности.

Несмотря на то, что у нас есть «глобальная» система координат (X,Y), каждую из задач удобнее решать в свой локальной системе координат. Каждая из задач в своей локальной системе координат может иметь свои собственные уравнения движения, число которых может меняться от задачи к задаче. Полет имеет непрерывную траекторию, поэтому в момент смены поведения должны выполняться условия непрерывности.

Модель подъема. Подъем в нашем случае влияет лишь на время аттракциона. Движение материальной точки (акробата), и это первое упрощение, можно описать как дискретное изменение ее высоты y1 (например, на одну единицу высоты в единицу модельного времени). Условием окончания подъема можно считать условие $(y1 \geq L)$. Таким образом, поведение — это решение разностного уравнения первого порядка $y_n^1 = y_{n-1}^1 + h_{\text{stair\_step}};\ \ y_0^1 = 0;\ \ Y = y^1;\ \ X = 0$, представленного в виде кусочно-постоянной функции, как функции от непрерывного времени.

Модель скольжения по прямой. Выберем локальную систему координат, как показано на рис. 2.16.

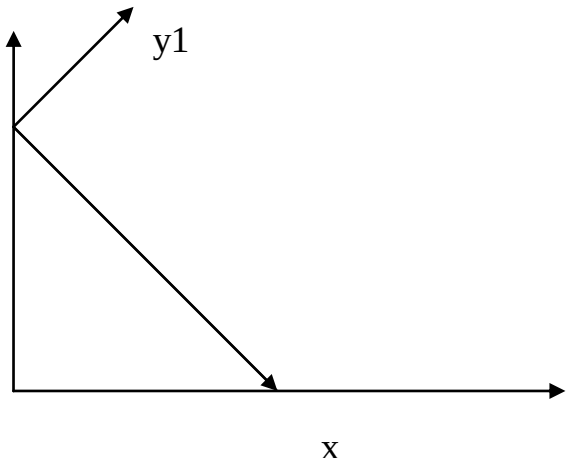


Рис. 2.16. Локальная система координат, горизонтальная ось которой совпадает с линией скольжения

В этой системе координат достаточно написать уравнения движения только вдоль горизонтальной оси. На точку действуют сила тяжести $P = m \cdot g$, нормальная реакция наклонной плоскости N, сила сопротивления R, например, пропорциональная квадрату скорости $R = -k^2 m v^2$. Уравнение движение вдоль оси x1 запишется в виде

$$\frac{d^2 x1}{dt^2} = g \cdot \sin(\alpha) - k^2 \frac{dx1}{dt}.$$

Уравнение движение вдоль оси y1 выглядит так

$$m\frac{d^2y1}{dt^2} = 0 = N - m \cdot g \cdot \cos(\alpha),$$

что позволяет найти реакцию $N = m \cdot g \cdot \cos(\alpha)$.

Модель движения по отрезку окружности. Выберем локальные оси как показано на рис. 2.17.

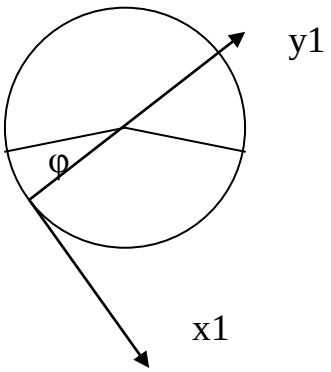


Рис. 2.17. Движение по дуге

Введем новую систему координат, у которой ось y1 направлена по радиусу окружности, а ось x1 совпадает с касательной к окружности в точке, где находится акробат. Угол φ показывает текущее положение радиуса, отсчитанного от начального положения. Проекция скорости на ось x1 равна $V_{x1} = r \cdot \frac{d\varphi}{dt}$, на акробата действует его сила тяжести и нормальная реакция. Уравнение движения может быть записано в виде $m \cdot \frac{dV_{x1}}{dt} = mg\sin(\alpha - \varphi)$. Для квадрата скорости справедливо уравнение $m \cdot \frac{1}{r} \cdot \left(\frac{dv}{dt}\right)^2 = N - mg\cos(\alpha - \varphi)$. Переписав уравнения относительно φ и учитывая, что движение происходит по окружности, получаем

$$r \cdot \frac{d^2\varphi}{dt^2} = g \cdot \sin(\alpha - \varphi),$$

$$N = m \cdot \left(\frac{d\varphi}{dt}\right)^2 - g \cdot \cos(\alpha - \varphi).$$

В этих уравнениях предполагается, что движение происходит без трения.

Свободный полет. В данном случае стоит вернуться в декартову систему координат (x1,y1), совместив начало отсчета по оси x1 с началом бассейна.

Уравнения свободного полета без сопротивления воздуха.

$$\frac{d^2x1}{dt^2} = 0,$$

$$\frac{d^2y1}{dt^2} = -g.$$

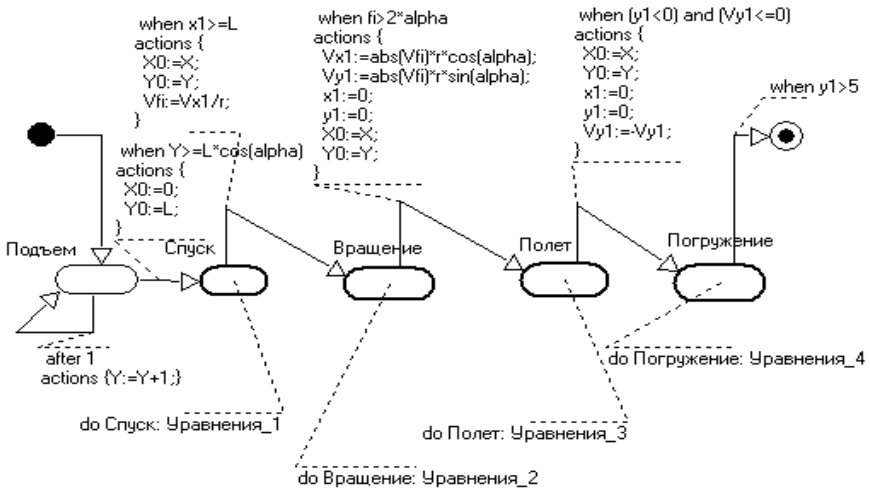


Рис. 2.18. Карта поведения модели «Аттракцион»

Погружение. Уравнения движения запишем в той же системе координат, что и при свободном полете, только вертикальную ось направим вглубь бассейна и учтем сопротивление воды, пропорциональное скорости

$$\frac{d^2x1}{dt^2} = -k\frac{dx1}{dt},$$
$$\frac{d^2y1}{dt^2} = g - -k\frac{dy1}{dt}.$$

Карта поведения для модели аттракциона показана на рис. 2.18, временная и фазовые диаграммы — на рис. 2.19 и рис. 2.20.

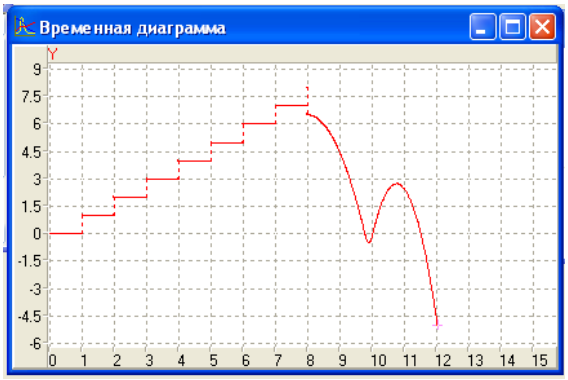


Рис. 2.19. Временная диаграмма в глобальных осях

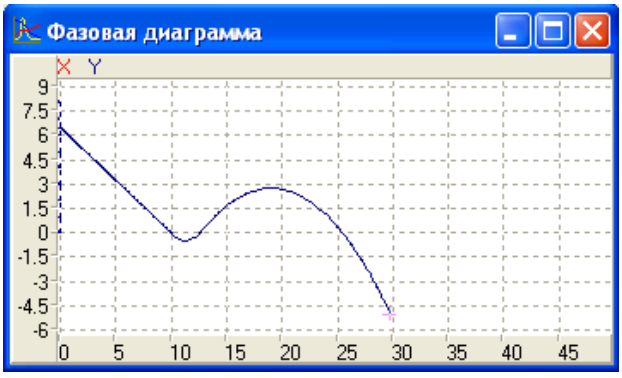


Рис. 2.20. Фазовая диаграмма в глобальных осях

Данная карта демонстрирует давно уже применяющийся в механике метод припасовывания, когда сложная задача разбивается на последовательные более простые, и их решения «сшиваются» на границах. Мы нигде здесь не выходим за рамки классической механики, и гибридные система является лишь удобным обобщением классических динамических систем. Заметим, что все написанные уравнения имеют решение в квадратурах (это синоним «решения в замкнутой форме»), и данную задачу можно рассматривать как тестовую. В то же время стоит добавить еще один трюк, а именно, подобрать скорость свободного полета так, чтобы вместо погружения можно было бы оттолкнуться от воды и пролететь еще несколько метров над водой, и мы вынуждены будем применять в качестве модели гибридную систему, так как удар (отражение от воды) — это мгновенное, скачкообразное изменение скорости.

3. МНОКОМПОНЕНТНЫЕ СИСТЕМЫ С ВХОДАМИ–ВЫХОДАМИ

Задания этого раздела предназначены для части «Компонентное моделирование сложных динамических систем». Название раздела указывает на тип связей, используемых при формировании структурной схемы модели.

Задания 1 – 6 следует использовать как лабораторные работы при изучении темы «Объектно-ориентированное моделирование. Классы, экземпляры. Инкапсуляция, наследование. Внешние переменные типа «входы-выходы». Компоненты с входами-выходами характерны для задач теории управления. В приведенных заданиях информация о компоненте передается направленно — от объекта управления (положение, скорость) к управляющему (следящему объекту), обрабатывается и возвращается в виде команд (сигналов) управления. В данных заданиях собственно управление сведено к минимуму, что позволяет сосредоточиться на изучении языковых конструкций, применяемых для задач этого типа. Эти лабораторные работы рекомендуется выполнять как первый, подготовительный этап сложной курсовой работы.

Задания 7 – 18 — это двенадцать однотипных (по сложности и методике решения) задач, которые могут использоваться либо как курсовая работа, либо как курсовой проект.

Сначала строится описание проектируемой модели на языке UML. Затем это описание используется для построения моделей в двух различных средах моделирования, например, Simulink и RMD. Полученные результаты сравниваются.

В отчете должны быть представлены:

описание системы в терминах унифицированного языка моделирования UML; модель системы, реализованная в подсистеме Simulink пакета MATLAB на основе UML-описания; модель системы, реализо-

ванная в пакете Studium на основе UML-описания; результаты вычислительного эксперимента и их объяснение.

При построении модели любой из задач в пакете RMD необходимо реализовать трехмерную анимацию (задания 7 – 16) или создать панель управления параметрами, содержащую в себе компоненты двумерной анимации (задания 17 – 18).

ЗАДАНИЕ 1

Самолет летит горизонтально на высоте H с постоянной скоростью Vairplane = 10 м/с. Когда он достигает точки, удаленной от начала координат на расстоянии S, начинается двигаться шар со скоростью Vball = 20 м/с под углом Alpha (рис. 3.1).

Найдите, когда шар и самолет столкнутся, используя приведенные ниже соотношения:

$$x_{airplane} = x_{ball},\ x_{airplane} = V_{airplane} * t_{\text{collision}},$$

$$x_{ball} = V_{ball} * \sin(Alpha) * t_{\text{collision}},\ y_{airplane} = y_{ball},\ y_{airplane} = H,$$

$$y_{ball} = V_{ball} * \cos(Alpha) * t_{\text{collision}} - \frac{g t^2_{\text{collision}}}{2},\ g = 9{,}81\ м^2/с.$$

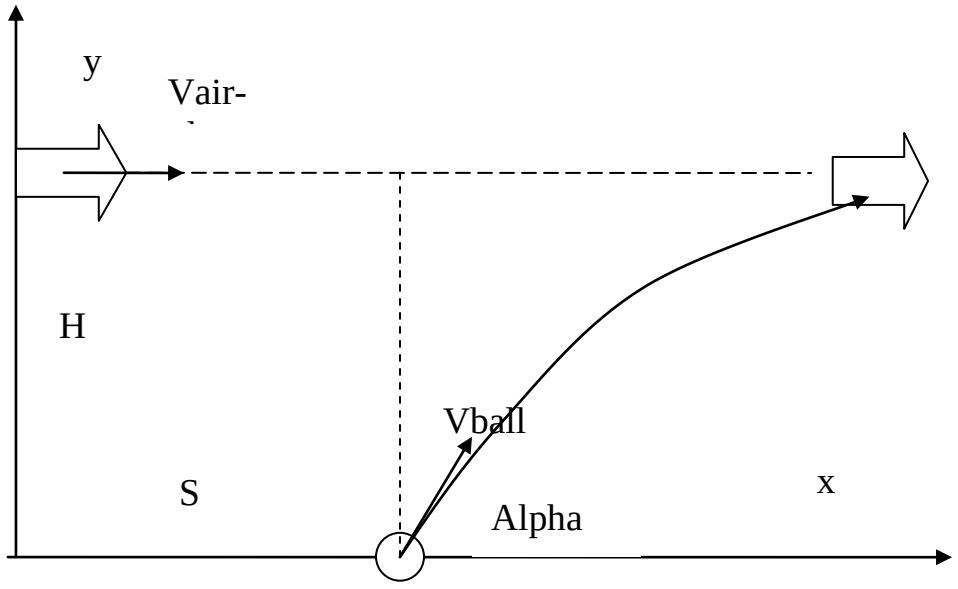


Рис. 3.1. Условие задания 1

Реализуйте это движение в виде игры, когда нужно экспериментально подбирать угол, обеспечивающий столкновение.

ЗАДАНИЕ 2

Самолет движется горизонтально на высоте H=10 м с постоянной скоростью Vairplane = 10 м/с. На расстоянии S от начала координат, от него отделяется шар и начинает свободно падать (рис. 3.2). Чему равно значение S, если известно, что шар приземлился на расстоянии Lx = 20 м?

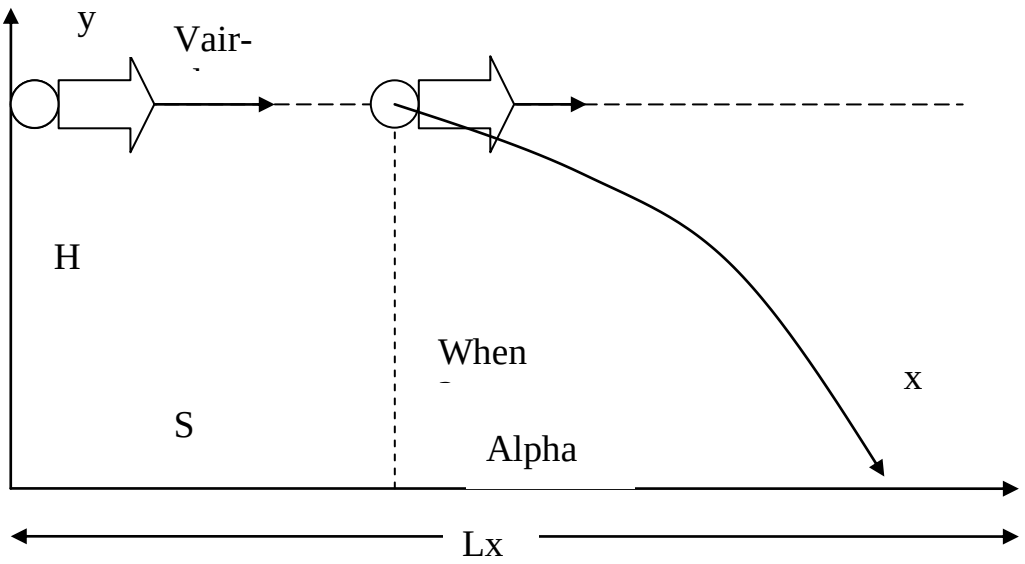


Рис. 3.2. Условие задания 2

Используйте приведенные ниже формулы:

$$S = V_{aiplane} \cdot t_x,$$

$$H = \frac{g \cdot t^2_{fall}}{2}; t_{fall} = \sqrt{\frac{2H}{g}}; Lx = V_{aiplane} \cdot (t_x + t_{fall}),$$

$$g = 9{,}81\, м^2 / с.$$

Реализуйте следующую игру. Пользователь следит за движением самолета и сам выбирает момент отделения шара, для того чтобы попасть в заданную точку.

ЗАДАНИЕ 3

Самолет движется горизонтально на высоте H = 10 м с постоянной скоростью Vairplane = 10 м/с. В момент времени t = 0 (x = 0) от самолета отделяется шар и начинает свободно падать, касается поверхности и отскакивает (рис. 3.3). На каком расстоянии Lx шар вторично коснется земли, если считать столкновение абсолютно упругим? Создайте игру, в которой можно менять значение H и фиксировать время второго касания. Как зависит время касания от высоты H?

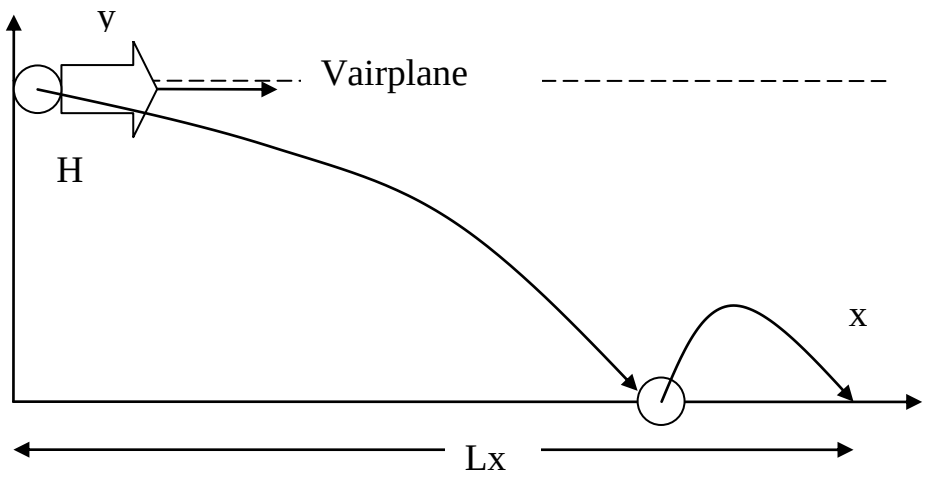


Рис. 3.3. Условие задания 3

ЗАДАНИЕ 4

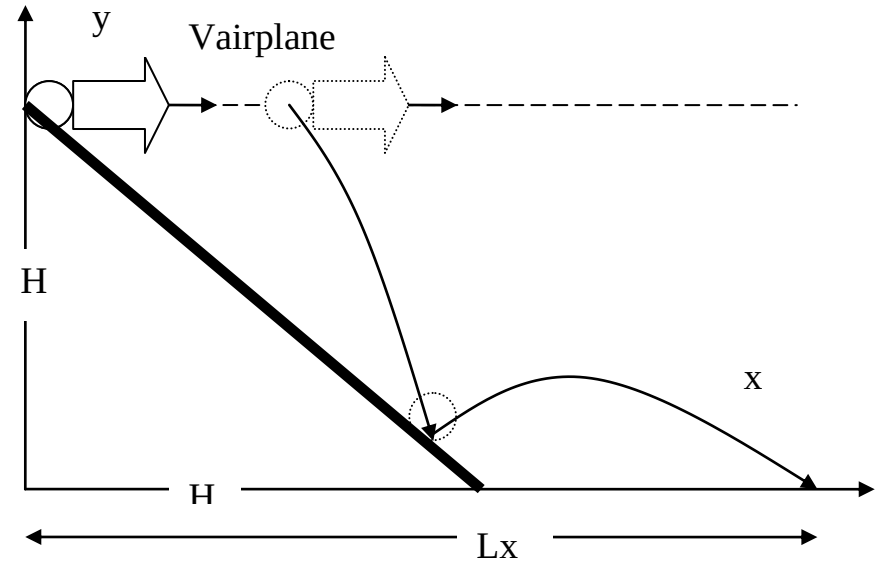


Рис. 3.4. Условие задания 4

Самолет с шаром на борту летит на высоте H = 20 м с постоянной скоростью Vairplane = 10 м/с. Через одну секунду полета t = 1 (с) шар отделяется от самолета и начинает свободное падение (рис. 3.4). Попадет ли шар на наклонную плоскость? Если попадет, на каком расстоянии от начала координат приземлится? Отскок считать абсолютно упругим. Предусмотреть в игре возможность менять H.

ЗАДАНИЕ 5

На плоскости расположен прямоугольник со сторонами Sx = 10 и Sy = 20. В начальный момент времени шар начинает движение с постоянной скоростью Vball = 10 м/с из точки (x = 0, y = Sy/2) (рис. 3.5). При каком угле Alpha реализуется нарисованная траектория? Создайте игу, в которой пользователь может менять угол и начальную скорость, а также подбирать их экспериментально.

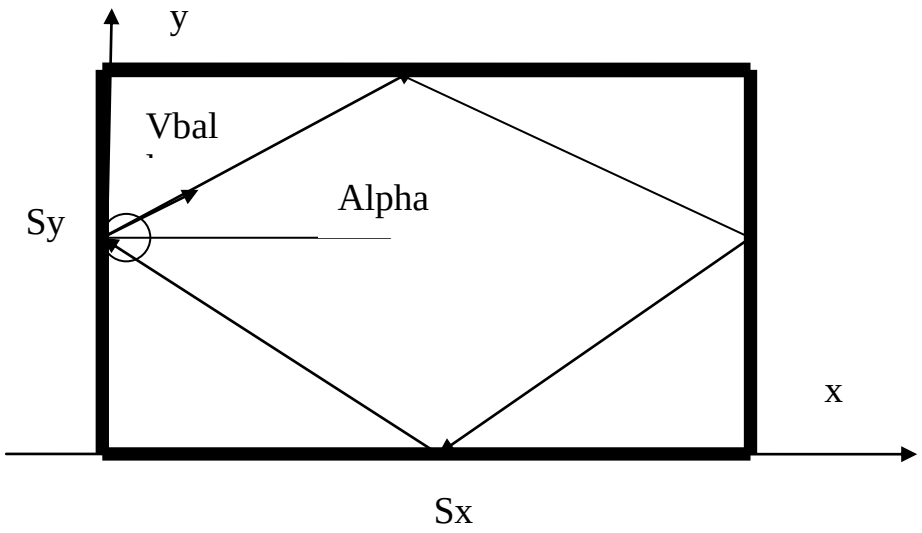


Рис. 3.5. Условие задания 5

ЗАДАНИЕ 6

На плоскости лежит прямоугольный треугольник со сторонами Sx = 20 и Sy = 20. Упругий шар начинает движение с постоянной скоростью Vball = 10 м/с. Начальное положение шара (x = 0, y = Sy/2) (рис. 3.6). При каком значении угла Alpha надо начать движение, чтобы попасть в точку (x = Sx/2, y = 0)?

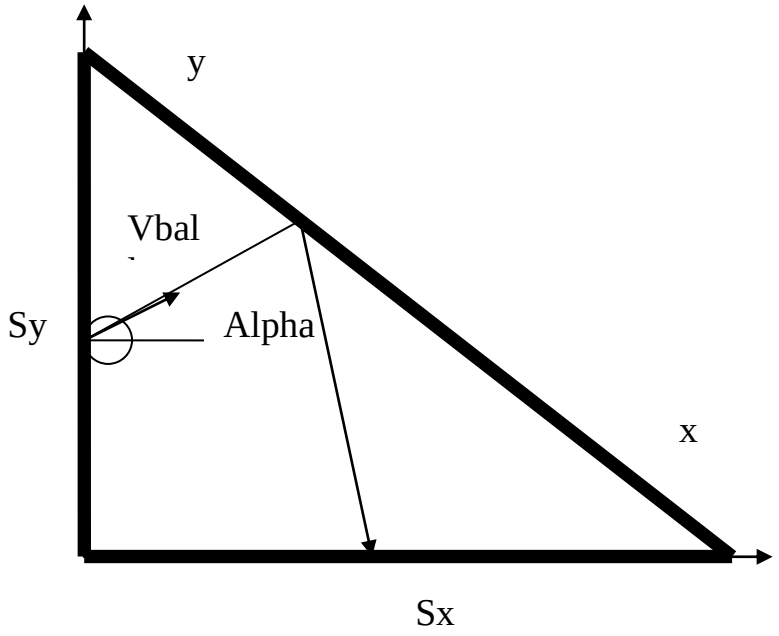


Рис. 3.6. Условие задания 6

Реализуйте игру, в которой можно менять угол и начальную скорость. Как построить периодическое движение?

ЗАДАНИЕ 7

Все последующие задания этого раздела заимствованы из книги [3].

Переменная сила Q передается на фундамент машиной с неуравновешенным ротором (рис. 3.7).

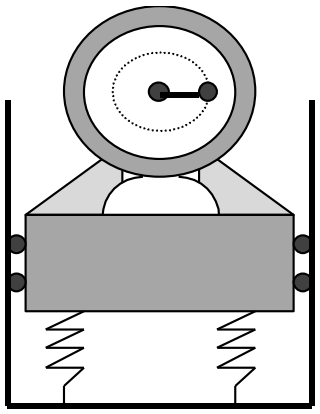


Рис. 3.7. Машина с неуравновешенным ротором

Сила Q меняется следующим образом: через каждые 50 секунд один вид гармонических колебаний

$$Q_1 = H_1 \cdot \sin(\omega_1 t) \tag{3.1}$$

сменяет другой

$$Q_2 = H_2 \cdot \cos(\omega_2 t), \tag{3.2}$$

где H_1 и H_2 — амплитуда возмущающей силы; ω_1 и ω_2 — частота возмущающей силы; $H_1 = 2$ см, $H_2 = 3$ см, $\omega_1 = 0{,}3$ с$^{-1}$, $\omega_2 = 0{,}5$ с$^{-1}$.

Если q — положение центра тяжести машины, то уравнения для различных возмущающих сил запишутся в виде:

$$\ddot{q} + k^2 \cdot q = \frac{Q_1}{a} \tag{3.3}$$

для гармонического закона (3.1),

$$\ddot{q} + k^2 \cdot q = \frac{Q_2}{a} \tag{3.4}$$

для гармонического закона (3.2), где $a = 1$ с$^{-2}$ — инерционный коэффициент; $k = 0{,}4$ с$^{-2}$ — частота собственных колебаний машины.

Начальные условия: $q|_{t=0} = 0$ и $\dot{q}|_{t=0} = 0$.

При таком воздействии на систему величина q может превзойти критическую величину $L_k = 100$, и машина выйдет из строя. Предположим, что мы можем управлять частотами возмущающей силы ω_1 и ω_2. Проверить экспериментально, можно ли для избегания аварии, при превышении q по абсолютному значению величины $L = 50$, уменьшать частоты на 10 %. Процесс изменения частоты требует 10 секунд модельного времени.

Можно ли таким способом управлять двумя машинами с разными значениями коэффициентов $k_1 = 0{,}4$ и $k_2 = 0{,}5$; $a_1 = 1$ и $a_2 = 0{,}5$. Машины следует «запускать» поочередно — сначала первую, а затем через 10 секунд модельного времени — вторую.

ЗАДАНИЕ 8

Дана система из двух маятников — маятника 1 и обращенного маятника 2 (рис. 3.8).

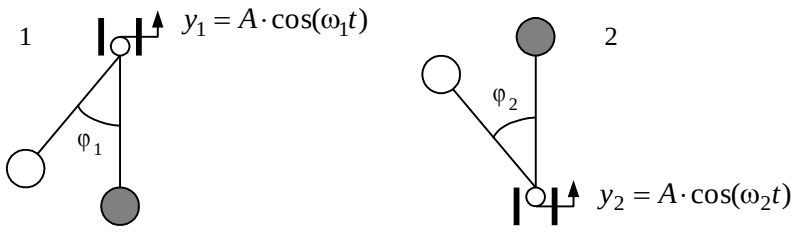


Рис. 3.8. Система маятников

Точка подвеса первого маятника гармонически колеблется по вертикали около своего среднего положения по закону (3.5), а точка подвеса второго маятника — у своего по закону (3.6):

$$y_1 = A \cdot \cos(\omega_1 t), \tag{3.5}$$

$$y_2 = A \cdot \cos(\omega_2 t), \tag{3.6}$$

где A — амплитуда колебаний; ω_1 и ω_2 — частоты колебаний, $A = 2$ м, $\omega_1 = 3$ с$^{-1}$, $\omega_2 = 3$ с$^{-1}$.

При этом каждые 10 секунд значение A изменяется на величину $\pm 0{,}5$ поочередно. Длина каждого из маятников равна $l = 40$ м.

Пусть φ_1 — угол отклонения первого маятника от строго вертикального положения, а φ_2 — угол отклонения второго маятника от строго вертикального положения. У первого маятника φ_1 изменяется в соответствии с уравнением:

$$\ddot{\varphi}_1 + \left(\frac{g}{l} + \frac{A\omega_1^2}{l} \cdot \cos(\omega_1 t) \right) \cdot \varphi_1 = 0. \tag{3.7}$$

Второй маятник колеблется по закону

$$\ddot{\varphi}_2 + \left(\frac{g}{l} + \frac{A\omega_2^2}{l} \cdot \cos(\omega_2 t) \right) \cdot \varphi_2 = 0, \tag{3.8}$$

и если $A\omega > \sqrt{2gl}$, где g — ускорение свободного падения, то маятник со временем приходит в устойчивое состояние $\varphi_2 = 0$.

У обоих маятников в начальный момент времени отклонение от положения равновесия составляет 0,1°. Значения φ_1 и φ_2 изменяются в пределах от 0 до 360°.

Создайте модель системы и убедитесь, что второй маятник приходит в устойчивое положение при этих значениях параметров.

Предположим теперь, что движение точек подвеса осуществляется устройством, которое не может работать постоянно и вынуждено останавливаться через каждые 10 секунд модельного времени на 1 секунду (т. е. в этот момент точки подвеса не двигаются и $\omega_1 = \omega_2 = 0$). Будет ли сохраняться устойчивое положение второго маятника в этом случае, и как изменится поведение первого?

Во всех экспериментах сначала начинает колебаться первый маятник, а через 3 секунды модельного времени — второй.

ЗАДАНИЕ 9

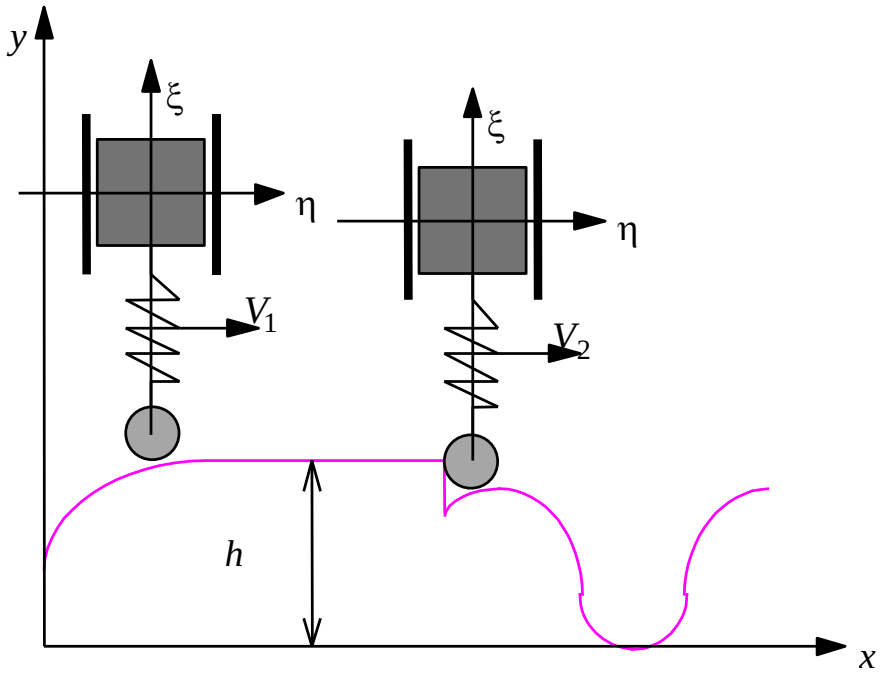


Рис. 3.9. Система «подрессорный груз – колесо»

Дан неровный участок со сложным профилем, который периодически повторяется (рис. 3.9). По нему движутся две идентичные «машины», каждая из которых представляет собой подрессорный груз

массой $m = 0{,}7$ кг, прикрепленный к безотрывно движущемуся с постоянной горизонтальной скоростью колесу. Скорость первой машины $V_1 = 0{,}9$ м/с, второй — $V_2 = 1$ м/с.

Профиль участка для одного периода описывается уравнениями

$$y = h \cdot (1 - e^{-\gamma x}),\ 0 \le x \le 1, \tag{3.9}$$

$$y = A_1 \cdot \cos(x),\ A_1 = h \cdot (1 - e^{-\gamma}),\ 1 < x \le 2, \tag{3.10}$$

$$y = A_2 \cdot \sin(x),\ A_2 = A_1 \cdot \cos(2),\ 2 < x < 4, \tag{3.11}$$

где h — высота горизонтального участка; γ — параметр, характеризующий кривизну профиля; A_1 и A_2 — амплитуды колебаний, $h = 1$ м, $\gamma = 1$.

Дифференциальные уравнения, описывающие вертикальные колебания ξ (в начальный момент времени отсутствующие) центра тяжести машины, движущейся с постоянной скоростью $x = Vt$, записываются в следующем виде:

- для первого участка:

$$\ddot{\xi} = -\frac{c}{m} \cdot \xi + h \cdot \gamma^2 \cdot V^2 \cdot e^{-\gamma V t}, \tag{3.12}$$

- для второго:

$$\ddot{\xi} = -\frac{c}{m} \cdot \xi + A_1 \cdot V^2 \cdot \cos(Vt), \tag{3.13}$$

- для третьего:

$$\ddot{\xi} = -\frac{c}{m} \cdot \xi + A_2 \cdot V^2 \cdot \sin(Vt), \tag{3.14}$$

где c — жесткость упругой подвески; $c = 0{,}5$ кг/с^2.

Первая машина в начальный момент времени расположена в начале пути, а вторая расположена в начале косинусоидального участка, и они начинают движение со скоростями $V_1 = 12$ и $V_2 = 10$.

За движением машин следит система управления. Если расстояние между машинами становится меньше чем $L = 0{,}2$, то догоняющая машина снижает свою скорость на 50 %. Если расстояние оказывается

меньшим, чем $L_k = 0{,}1$, то считается, что произошла авария. Если расстояние между машинами больше $L_b = 1$, то скорость обеих машин увеличивается на 30 % от текущего значения.

Построить модель системы.

ЗАДАНИЕ 10

Жесткая плоская пластинка длиной $l = 5$ м находится в потоке газа (жидкости), скорость $V = 1$ м/с которого горизонтальна и направлена так, как показано на рис. 3.10.

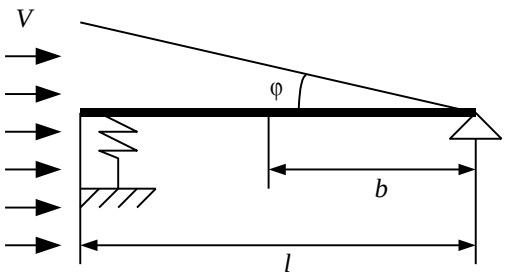


Рис. 3.10. Система «пластина – газ»

В этом положении аэродинамические силы равны нулю, и пластинка находится в равновесии под действием силы тяжести и реакции опор. При отклонении пластинки возникают аэродинамические давления, зависящие от угла отклонения φ.

Предположим, что в начальный момент мы отклонили пластинку от положения равновесия на угол 0,01°. Пусть $I = 1$ кг $\cdot$ м^2 — момент инерции пластинки относительно оси шарнира. Уравнения для угла можно записать в виде:

$$I \cdot \ddot{\varphi} + \left(c_0 \cdot l - k_y \cdot \frac{\rho V^2}{2} \cdot b \right) \cdot l \cdot \varphi = 0, \qquad (3.15)$$

где c_0 — коэффициент жесткости пружины; k_y — постоянный аэродинамический коэффициент; ρ — плотность газа; b — расстояние от оси шарнира, определяющее точку приложения равнодействующих

аэродинамических давлений на пластину, $c_0 = 0{,}5$ кг/с^2, $k_y = 0{,}5$ м $\cdot$ с^2, $\rho = 2$ кг/м^3, $b = 1$ м.

Если выполняется условие $c_0 \cdot l - k_y \cdot \dfrac{\rho V^2}{2} \cdot b > 0$, то в системе наблюдаются возрастающие по амплитуде колебания, в противном случае под действием аэродинамических сил пластинка со временем возвращается в положение равновесия.

Построить модель системы и убедиться, что при выбранных значениях параметров пластина с течение временем приходит в устойчивое состояние.

Предположим теперь, что в начальный момент пластина находится в состоянии равновесия. Каждые 5 секунд скорость подаваемого газа то увеличивается на 50 %, то возвращается к прежнему значению. Также каждые 10 секунд плотность газа то увеличивается на 50 %, то возвращается к прежнему значению. В некоторый случайный момент времени пластина мгновенно отклоняется на угол 0,01°. Будут ли при таких условиях наблюдаться аварийные ситуации, если аварийным считается отклонение пластины более чем на 45°?

Пусть в рассмотренных условиях пластина проработала 50 единиц модельного времени и вышла из строя. Ее мгновенно заменили новой, у которой конструктивные параметры c_0, k_y отличаются на 10 %. Будут ли при таких условиях наблюдаться аварийные ситуации?

ЗАДАНИЕ 11

Дана система из двух тел с массами m_1 и m_2, соединенных двумя пружинами, жесткости которых равны c_1 и c_2 (рис. 3.11), $m_1 = 1$ кг, $m_2 = 1$ кг, $c_1 = 1$ кг/с^2, $c_2 = 1$ кг/с^2.

На левый груз системы действует гармоническая возмущающая сила Q, которая через каждые 20 секунд меняет свой вид:

$$Q = \begin{cases} H_1 \cdot \sin(\omega t), \\ H_2 \cdot \cos(\omega t), \end{cases} \qquad (3.16)$$

где H_1 и H_2 — амплитуды колебаний, ω — частота колебаний, $H_1 = 1$ м, $H_2 = 1{,}5$ м, $\omega = 2$ с$^{-1}$.

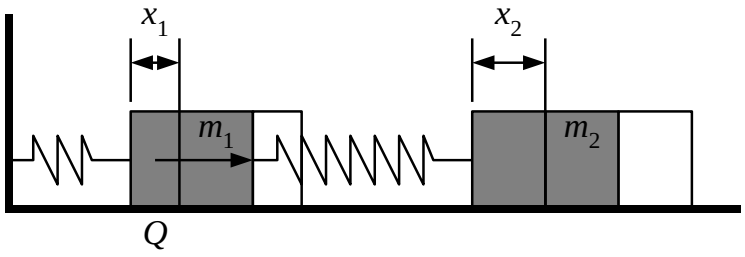


Рис. 3.11. Система двух грузов на пружинах

Пусть x_1 и x_2 — горизонтальное отклонение грузов от положения равновесия (в начальный момент времени отсутствующее). Тогда уравнениями движения будут:

$$m_1\ddot{x}_1 + (c_1 + c_2)\cdot x_1 - c_2 x_2 = Q, \qquad (3.17)$$

$$m_2 x_2 - c_2 x_1 + c_2 x_2 = 0. \qquad (3.18)$$

Построить модель данной системы, создав для каждого груза свой отдельный блок с уравнениями (3.17) и (3.18).

Предусмотреть возможность изменять частоту возмущающей силы в произвольный момент на 25 % по желанию экспериментатора. Частота должна автоматически к исходному значению через 10 секунд модельного времени. В эти 10 секунд изменение частоты экспериментатором невозможно. При моделировании системы останавливать моделирование всегда, если расстояние между грузами стало меньше 10 % от своего первоначального значения L, равного $L = 3$ или $L = 5$.

ЗАДАНИЕ 12

Дан вертикальный невесомый упругий стержень длиной $l = 5$ м с постоянной жесткостью сечения $EJ = 1$ кг $\cdot$ м^3/с^2. С концом стержня связан сосредоточенный груз массой $m = 1$ кг. Верхней опорой стержня служит неподвижный шарнир, а нижней — подвижная втулка (рис. 3.12).

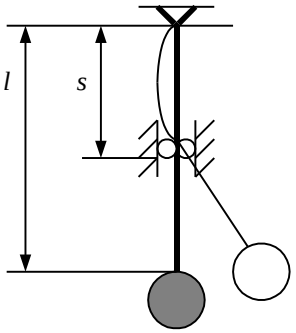


Рис. 3.12. Система
«стержень – груз»

Расстояние между опорами $s < l$ не постоянно и меняется с периодом 30 секунд по одному из двух гармонических законов:

$$s = \begin{cases} H_1 \cdot \cos(\omega_1 t), \\ H_2 \cdot \sin(\omega_1 t), \end{cases} \tag{3.19}$$

где H_1 и H_2 — амплитуды колебаний; ω_1 — частота колебаний, $H_1 = 1{,}5$ м, $H_2 = 2$ м, $\omega_1 = 1$ с$^{-1}$.

Частота колебаний вне зависимости от закона колебаний каждые 20 секунд то уменьшается на 40 %, то возвращается к прежнему значению.

Состоянию равновесия соответствует положение груза на вертикали и прямолинейная форма оси стержня. При возмущении этого состояния груз отклоняется в сторону, ось балки изгибается, и последующее движение описывается дифференциальным уравнением:

$$\ddot{x} + \frac{3 \cdot EJ}{m \cdot l \cdot [l - s]^2} \cdot x = 0. \tag{3.20}$$

В начальный момент времени происходит отклонение груза на расстояние $x_0 = 0{,}1$ м. Если x превышает пороговое значение $x_{\max} = 3$ м, то стержень разрушается.

Построить модель данной системы и проверить, если в ней аварийные режимы.

Создать виртуальный тренажер, в котором в начальный момент времени стержень находится в состоянии равновесия. В случайные моменты времени он может мгновенного отклоняться на расстояние

от 0,1 до 0,2 м. Задача оператора — отслеживать аварийные ситуации и в случае их возникновения мгновенно менять вышедшее устройство на новое, конструктивные параметры которого l, m, EJ не более чем на 10 % отличаются от исходных.

ЗАДАНИЕ 13

Дана система из груза массой $m = 1$ кг, связанного с невесомой жесткой упруго закрепленной балкой.

Пусть $2l$ — длина балки, c_0 — коэффициент жесткости пружины, $l = 1$ м, $c_0 = 1$ кг/с^2. Один конец балки закреплен на шарнире, расположенном на неподвижной опоре (рис. 3.13).

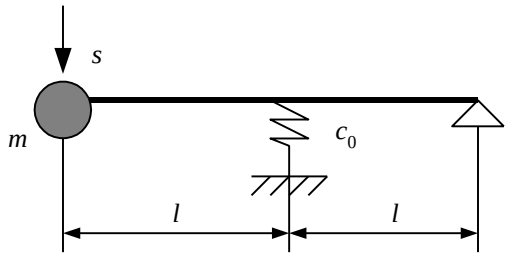


Рис. 3.13. Система
«груз – балка – пружина»

В начальный момент времени происходит однократный вертикальный удар по грузу с величиной мгновенного ударного импульса $S = 2$ Н · с. Скорость груза получает мгновенное приращение:

$$\varphi|_{t=0} = 0,$$

$$\dot{\varphi}|_{t=0} = \frac{S}{2 \cdot m \cdot l},$$

где φ — это угол отклонения системы от положения равновесия.

Движение системы описывается следующим уравнением:

$$4 \cdot m \cdot \ddot{\varphi} + c_0 \cdot \varphi = 0. \qquad (3.21)$$

Через 25 секунд после начала колебаний масса груза m мгновенно уменьшается на 50 %. Далее движение системы продолжается, но уже с грузом новой массы.

Если угол φ становится больше предельного значения $\varphi_{\max} = \dfrac{S}{\sqrt{c_0 ml}}$, то происходит разрушение системы.

Построить модель системы и проверить экспериментально, наступают ли в ней аварийные режимы при заданных значениях параметров.

Построить тренажер, состоящий из двух таких систем, масса которых может меняться с помощью пульта управления экспериментатором, а конструктивные параметры l и $c_0 = 1$ меняются случайным образом не более чем на 10 % в начале каждого нового эксперимента. Каждый эксперимент продолжается 100 секунд модельного времени, в течение которых в случайные моменты времени происходит удар то по одному грузу, то по другому (т. е. начальные условия выбираются по формуле (3.20)). Эксперимент считается удачным, если ни в одной из подсистем значение угла не превзошло критического значения.

ЗАДАНИЕ 14

В воде плавает кусок пробки в форме параллелепипеда с площадью основания $S = 1$ м^2 и высотой $H = 0{,}5$ м. Пробку погружают в воду на небольшую глубину $x_0 = 5$ см и отпускают (рис. 3.14).

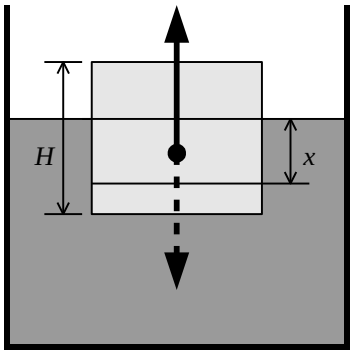


Рис. 3.14. Система «вода – пробка»

В результате пробка начинает совершать колебания. Сопротивление воды не учитывается. Изменение глубины погружения пробки в воду x описывается уравнением:

$$\ddot{x} + \frac{\rho_в \cdot g}{\rho_п \cdot H} \cdot x = 0 \qquad (3.22)$$

с начальными условиями:

$$x\big|_{t=0} = x_0;\ \dot{x}\big|_{t=0} = 0,$$

где $\rho_в = 1000$ кг/м^3 — плотность воды; $\rho_п = 200$ кг/м^3 — плотность пробки; g — ускорение свободного падения.

Пусть существует вторая ванна с пробкой, но в ней в начальный момент пробке, плавающей на поверхности, сообщили скорость, равную $v_0 = 1$ м/с. При изучении поведения пробки во второй ванне учли сопротивление воды, положив силу сопротивления равной $F_с = -r \cdot v$, где $r = 1$ кг/с — коэффициент пропорциональности. Колебания пробки в этой системе описываются уравнением:

$$\ddot{x} + \frac{r}{m} \cdot \dot{x} + \frac{\rho_в \cdot g}{\rho_п \cdot H} \cdot x = 0 \qquad (3.23)$$

с начальными условиями:

$$x\big|_{t=0} = 0;\ \dot{x}\big|_{t=0} = v_0,$$

где m — масса пробки.

Построить модель системы и экспериментально выяснить, какая из пробок скачет выше.

Построить также модель системы, в которой каждые 20 секунд в первой системе вода мгновенно «превращается» в ртуть, а ртуть с той же периодичностью — в воду, а во второй системе каждые 25 секунд происходят аналогичные «превращения» воды в спирт. Плотность ртути — $\rho_р = 1360$ кг/м^3, плотность спирта — $\rho_с = 790$ кг/м^3.

ЗАДАНИЕ 15

Материальная точечная масса $m = 1$ кг находится в поле тяготения тонкого кольца массы $M = 10^{20}$ кг и радиуса $R = 1$ км. В начальный момент времени материальная точечная масса помещается в точку Q_1, расположенную на оси кольца на расстоянии $x_0 < R$ от плоскости кольца и начинает совершать колебательные движения (рис. 3.15).

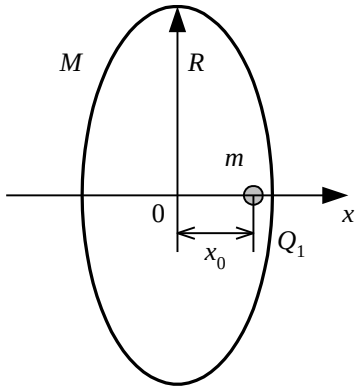


Рис. 3.15. Система «точечная масса – кольцо»

Если $x_0 < 0{,}1R$, то колебания x описываются следующим уравнением:

$$m \cdot \ddot{x} = -\frac{G \cdot M \cdot m}{R^3} \cdot x. \qquad (3.24)$$

Если $x_0 \geq 0{,}1R$ или в результате колебаний выполняется условие $x \geq 0{,}1R$, то колебания описываются следующим уравнением:

$$m \cdot \ddot{x} = -\frac{G \cdot M \cdot m}{\left(R^2 + x^2\right)^{3/2}} \cdot x, \qquad (3.25)$$

где G — гравитационная постоянная.

Пусть $x_0 = 1$ м.

Каждые 15 секунд радиус кольца поочередно мгновенно расширяется и сжимается в 10 раз. Если точка отклоняется от кольца на расстояние, превышающее в 10 раз его первоначальный радиус, система разрушается.

Построить модель данной системы, а также модель системы, состоящей из двух систем «материальная точка – кольцо», несвязанных друг с другом. Вторая система «материальная точка – кольцо» идентична первой, за исключением того, что в ней радиус кольца изменяется в 5 раз каждые 20 секунд.

ЗАДАНИЕ 16

В длинной вертикальной цилиндрической трубке, закрытой с нижнего конца, может ходить без трения поршень, масса $m = 10$ кг которого велика по сравнению с массой идеального газа, заключенного внутри трубки. В положении равновесия расстояние между поршнем и дном трубки равно $l_0 = 1000$ м (рис. 3.16).

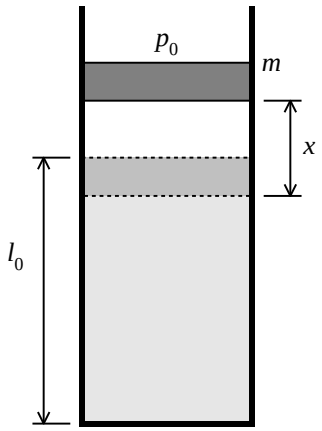


Рис. 3.16. Система «поршень – газ»

Площадь поперечного сечения трубки равна $S = 1$ м^2, на поршень действует нормальное атмосферное давление p_0. В начальный момент времени поршень отклоняют от положения равновесия на расстояние $x << l_0$. В результате этого поршень начинает совершать колебания, описываемые следующими уравнениями:

- если $p_0 \neq 0$, то

$$m \cdot \ddot{x} + \frac{m \cdot g + p_0 \cdot S}{l_0} \cdot x = 0, \qquad (3.26)$$

- если $p_0 = 0$, то

$$\ddot{x} + \frac{g}{l_0} \cdot x = 0, \tag{3.27}$$

где g — ускорение свободного падения.

Пусть $x = 1$ м.

Атмосферное давление каждые 10 секунд мгновенно изменяется с нормального значения на меньшее на 80 % (и обратно).

Построить модель данной системы, а также модель системы, состоящей из двух систем типа «трубка с поршнем», несвязанных друг с другом. Вторая «трубка с поршнем» идентична первой, за исключением того, что в ней поршень имеет другую массу $m' > m$ и другую площадь поперечного сечения трубки $S' > S$.

ЗАДАНИЕ 17

Дана однокамерная фармакокинетическая модель с всасыванием (рис. 3.17), где 1 — это место введения лекарственного препарата; 2 — камера.

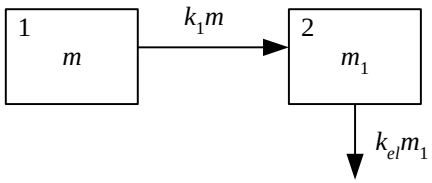


Рис. 3.17. Однокамерная фармакокинетическая модель с всасыванием

Камера представляет собой ограниченный в пространстве объем жидкости (ткани), неизменный с течением времени. Дан определенный объем лекарственного препарата, который всасывается в камеру пропорционально своей массе в соответствии с уравнением:

$$\frac{dm}{dt} = -k_1 \cdot m, \tag{3.28}$$

где m — масса лекарственного препарата в месте введения 1; k_1 — константа скорости поступления препарата в камеру (константа скорости всасывания).

Пусть $k_1 = 0{,}3\ \text{с}^{-1}$.

Предполагается, что масса лекарственного препарата в месте введения в начальный момент времени равна $m_0 = 30$ мг, причем в самой камере в начальный момент времени препарата нет. Тогда масса лекарственного препарата в камере изменяется в соответствии со следующим уравнением:

$$\frac{dm_1}{dt} = k_1 \cdot m - k_{el} \cdot m_1, \qquad (3.29)$$

где m_1 — масса лекарственного препарата в камере; k_{el} — константа элиминации (выведения) лекарственного препарата из камеры.

Пусть $k_{el} = 0{,}5\ \text{с}^{-1}$.

Как только масса лекарственного препарата в месте введения становится меньше порогового значения $\varepsilon = 0{,}001$ мг, засекается отрезок времени $Time = 10$ с, по истечении которого в месте введения уничтожаются все остатки прежней дозы препарата, и вводится новая доза $m = m_0$.

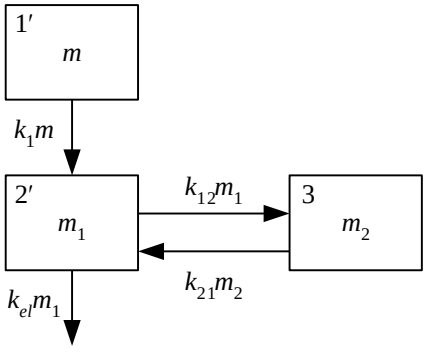


Рис. 3.18. Двухкамерная фармакокинетическая модель с всасыванием

Существует более сложная система, называемая двухкамерной фармакокинетической моделью с всасыванием (рис. 3.18).

В ней лекарственный препарат вводится из аналогичного ранее описанному места введения $1'$ и выводится из камеры $2'$, но существует еще камера 3, подсоединенная к камере $2'$. Между камерами $2'$ и 3 может циркулировать лекарственный препарат.

В этой модели масса лекарственного препарата в месте введения описывается также уравнением (3.28) с теми же начальными условиями. Массы же в камерах $2'$ и 3 описываются уравнениями (3.30) и (3.31) соответственно:

$$\frac{dm_1}{dt} = k_1 m - \left(k_{el} + k_{12}\right) \cdot m_1 + k_{21} m_2, \qquad (3.30)$$

$$\frac{dm_2}{dt} = k_{12} m_1 - k_{21} m_2, \qquad (3.31)$$

где m_2 — масса лекарственного препарата в камере 3; k_{12} — константа скорости поступления препарата из камеры $2'$ в камеру 3; k_{21} — константа выведения лекарственного препарата из камеры 3 в камеру $2'$.

Пусть $k_{12} = 0{,}4\ \text{с}^{-1}$, $k_{21} = 0{,}6\ \text{с}^{-1}$.

Как и в первой системе, в начальный момент времени в камерах $2'$ и 3 лекарственный препарат отсутствует. Как и в первой системе, как только масса лекарственного препарата в месте введения становится меньше порогового значения ε', засекается отрезок времени $Time_1$, по истечении которого в месте введения уничтожаются все остатки прежней дозы препарата, и вводится новая доза $m = m_0$.

Построить модель системы, содержащей обе фармакокинетические модели, несвязанные друг с другом с различными местами введения лекарственного препарата. Также построить модель системы, состоящей из обеих фармакокинетических моделей, имеющих общее место введения препарата. Скорость всасывания лекарственного препарата k_1 в обеих фармакокинетических моделях одинаковая, отрезок времени $Time'$ и пороговое значение ε' для места введения препарата

совпадают со значениями, принятыми для однокамерной фармакокинетической модели с всасыванием.

ЗАДАНИЕ 18

Даны две биологические популяции, оспаривающие одну и ту же пищу. Пусть это будут популяции медведей (численностью N_1) и волков (численностью N_2) (рис. 3.19).

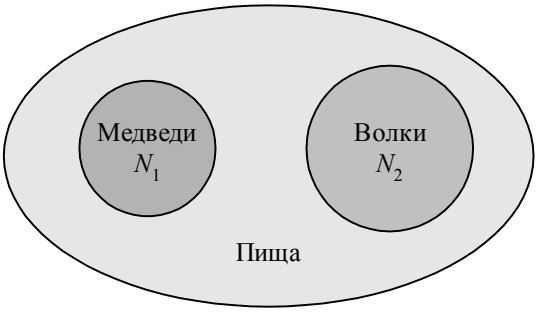


Рис. 3.19. Система двух биологических популяций

Пусть при количестве пищи, достаточном для полного удовлетворения рассматриваемых видов, существуют постоянные положительные коэффициенты прироста популяций: $\varepsilon_1 = 0{,}7$ мес$^{-1}$ для медведей и $\varepsilon_2 = 0{,}9$ мес$^{-1}$ для волков. Заданы «коэффициенты прожорливости»: $\gamma_1 = 0{,}7$ кг$^{-1}$ и $\gamma_2 = 0{,}5$ кг$^{-1}$, соответствующие потребности в пище для каждой из двух популяций.

Пусть $F(N_1, N_2)$ — количество пищи, поедаемой обеими популяциями в единицу времени. Оно задается уравнениями (3.32) и (3.33), где уравнение (3.32) соответствует случаю, когда обе популяции активны, а уравнение (3.33) — когда медведи впадают в спячку. Переключение между режимами (3.32) и (3.33) происходит периодически, причем с режима (3.32) на режим (3.33) переключение происходит через отрезок времени $Time_1 = 9$ месяцев, а с режима (3.33) на режим (3.32) — через $Time_2 = 3$ месяца:

$$F(N_1, N_2) = \lambda_1 \times N_1 + \lambda_2 \times N_2, \qquad (3.32)$$

$$F(N_1, N_2) = \lambda_2 \times N_2, \qquad (3.33)$$

где λ_1 и λ_2 — некие положительные коэффициенты.

Пусть $\lambda_1 = 0{,}01$ кг/(месяц · шт), $\lambda_2 = 0{,}02$ кг/(месяц · шт).

В начальный момент времени популяции обладают начальной численностью $N_1' = 10$ шт и $N_2' = 20$ шт.

Тогда развитие популяций описывается следующими уравнениями:

- медведи

$$\frac{dN_1}{dt} = \left[\varepsilon_1 - \gamma_1 \cdot F(N_1, N_2)\right] \cdot N_1, \tag{3.34}$$

- волки

$$\frac{dN_2}{dt} = \left[\varepsilon_2 - \gamma_2 \cdot F(N_1, N_2)\right] \cdot N_2. \tag{3.35}$$

Как только численность той или другой популяции становится меньше 1 (умирает последняя особь), засекается отрезок времени $Time_3 = 3$ мес (если это медведи) или $Time_3' = 5$ мес (если это волки), по истечении которого вместо прежней популяции поселяется новая популяция с новой начальной численностью N_1', если это медведи, и N_2' — если волки.

Построить модель данной системы, а также модель системы, состоящей из двух систем типа «две популяции», несвязанных друг с другом. Вторая система «две популяции» идентична первой, за исключением того, что в ней вместо медведей и волков в качестве конкурирующих видов рассматриваются лоси и олени, имеющие иные значения коэффициентов прироста ε_1' и ε_2', другие значения «коэффициентов прожорливости» γ_1' и γ_2', а также иные значения коэффициентов λ_1 и λ_2.

ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Система двух материальных точек. Рассмотрим механическую систему, состоящую из двух материальных точек P_1 и P_2 с массами M и m соответственно. Расположим эти точки на горизонталь-

ной плоскости (x, y) и позволим точке P_1 двигаться только вдоль оси y с постоянной скоростью V_y, а точке P_2 — вдоль оси x со скоростью V_x. На расстоянии L от начала координат на осях x и y установим четыре отражателя, при соприкосновении с которыми точки меняют направление движения на противоположное (рис. 3.20). Две материальные точки нарисованы в виде двух шаров разных диаметров. Шару с большим диаметром соответствует точка с большей массой.

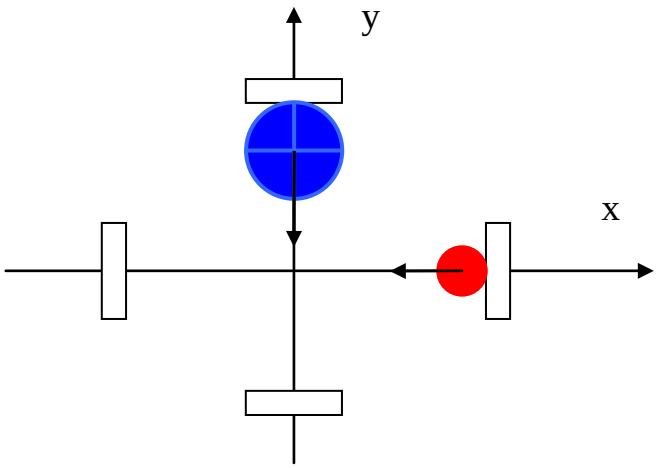


Рис. 3.20. Две материальные точки
на плоскости

Предположим, что масса M точки P_1 много больше массы m точки P_2. В этом случае при столкновении точек в начале координат точка с меньшей массой m меняет направление своего движения на противоположное, а точка с большей массой M продолжает двигаться в том же направлении, как будто бы ничего не случилось. В начальный момент точка P_1 имеет координаты $(0, L)$ и отрицательную скорость $V_y < 0$, то есть движется к началу координат, а точка P_2 — координаты $(L, 0)$ и отрицательную скорость $V_x < 0$.

При выбранном начальном расположении точек столкновения в начале координат возможны, если скорости V_y и V_x связаны между собой соотношением

$$|V_x| = k_x |V_y|, k_x = 1,3,5....$$

В системе двух материальных точек явно видны два компонента, взаимодействующих только при столкновении. Любая материальная точка меняет свой закон движения при достижении ограничителя (независимые события для любой точки) и при столкновении в начале координат (закон поведения точки с меньшей массой может измениться, если произойдет встреча с точкой с большей массой). Для модели в виде системы из двух компонентов со связью «по сигналу», обеспечивающей передачу сигнала о том, что шар с большей массой оказался в начале координат (рис. 3.21), карты поведения точек будут выглядеть так, как показано на рис. 3.22.

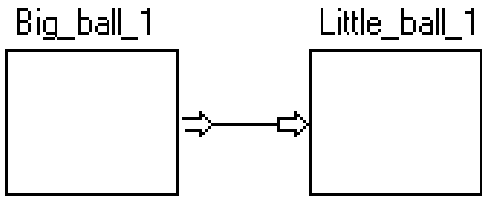


Рис. 3.21. Система двух шаров

Карты поведения для обоих шаров практически одинаковы. Переменные TwallX и TwallY нужны для вычисления времени достижения отражателей. При использовании явно заданного времени срабатывания перехода («AFTER») используются локальные часы — время отсчитывается от момента, когда состояние стало текущим. При срабатывании любого перехода, приводящего к возвращению в это же состояние (в нашем случае все переходы такие), локальные часы, отсчитывающие непрерывное время, запускаются заново, что приводит к необходимости создавать переменную $t' = (Time - NewTime) \equiv LocalTime$, где $Time$ — глобальное время, а $NewTime$ — время запуска локальных часов в глобальном времени, и пересчитывать времена наступления событий, приводящих к смене поведения. К их числу относится и время столкновения шаров Tcoll. В этот момент первый компонент Big_Ball посылает сигнал и про-

должает свою работу, а второй — Little_ball — принимает его и меняет свое поведение.

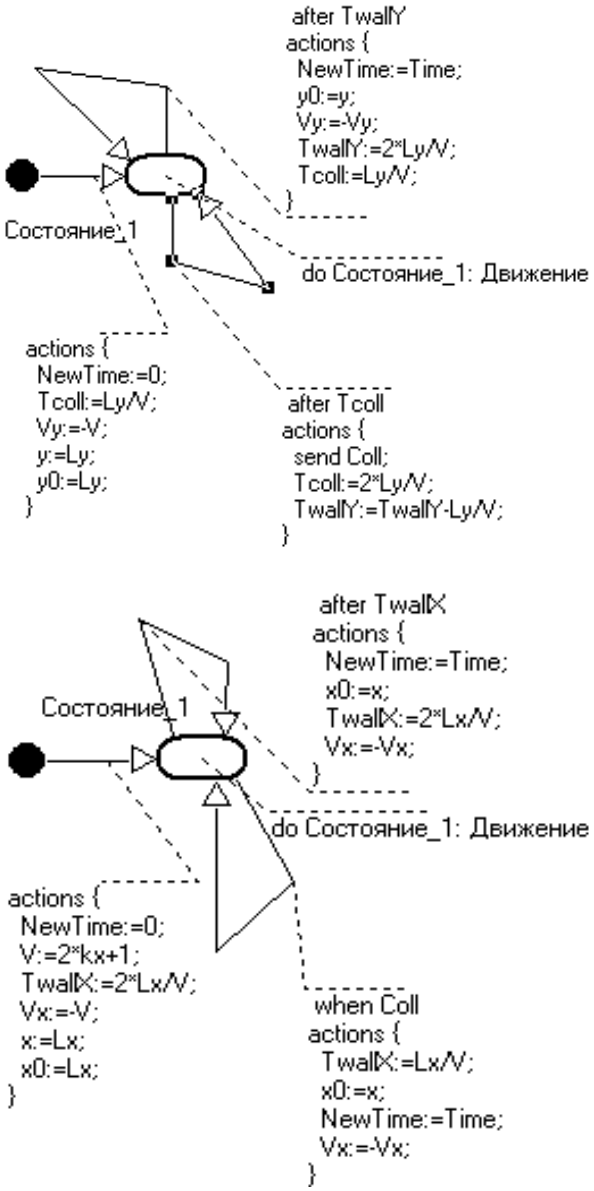


Рис. 3.22. Карты поведения
компонентов системы двух шаров

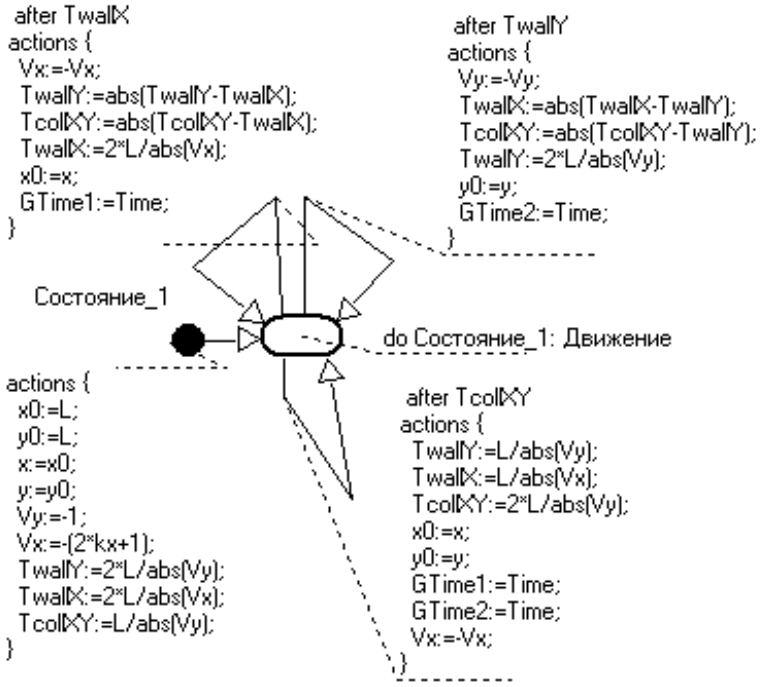


Рис. 3.23. Карта поведения системы двух шаров, представленных в виде единого объекта с двумя параллельными процессами

Можно ли использовать аналогичный прием построения карты поведения для системы двух шаров, представленных в виде единого объекта с двумя параллельными процессами одновременного движения шаров по соответствующим координатам? Да, можно. На рис. 3.23 представлена карта поведения такого объекта, аналогичная картам, приведенным на рис. 3.22.

Карту поведения на рис. 3.23 можно рассматривать как результат преобразования карт, приведенных на рис. 3.22, необходимого для реализации модели с несколькими параллельными процессами на однопроцессорной машине. На рис. 3.24 более заметна основная трудность сведения нескольких взаимодействующих параллельных процессов к одному последовательному. В нашей системе возможны два одновременных события — отражение шаров от своих отражателей. При реализации на однопроцессорной машине одновременные события должны стать последовательными, и порядок их обработки, то есть выполнение мгновенных действий с ними связанными, не дол-

жен влиять на результат. Они должны быть «перестановочны»! К счастью, для нашей модели это требование выполняется.

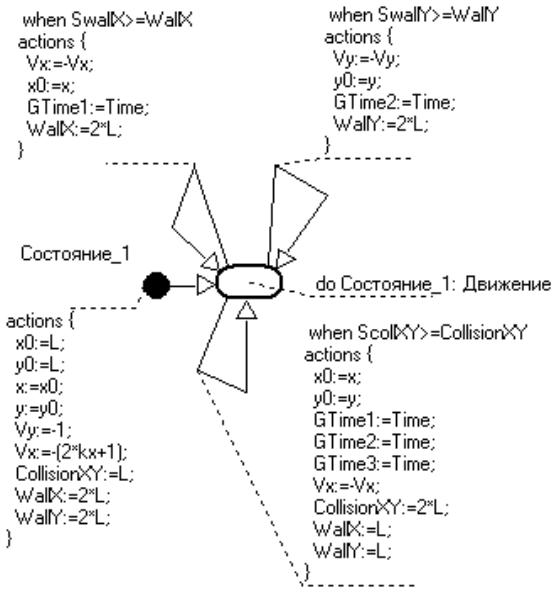


Рис. 3.24. Карта поведения,
использующая на «суть» событий

Приведенные выше карты поведения для системы и объекта опирались на информацию о временах событий («AFTER»), что приводило к необходимости явно использовать локальные часы. Более простым с точки зрения описания поведения является использование информации о сути события. На рис. 3.24 приведена карта поведения для единого объекта, опирающаяся информацию о сути событий. Обратите внимание, что в модели введены новые переменные — SwallX, WallX и т. д., путь шара по оси X и расстояние от начала координат по оси X, а также неравенства $SwallX \geq WallX$, позволяющие надежно определять, что произошло интересующее нас событие. Любые другие формы, например, для события «столкновение в начале координат» ($x = 0$ и $y = 0$) или $(abs(x) < \varepsilon$ и $abs(x) < \varepsilon)$ менее надежны или даже не реализуемы.

Пример двух материальных точек, независимо двигающихся между отражателями и сталкивающихся в начале координат, является

чрезвычайно наглядным примером, показывающим, чем отличаются (классические) динамические и гибридные системы.

В рассмотренной нами модели движения двух материальных точек между отражателями мы сталкиваемся с необходимостью выйти за рамки классической механики (динамических систем). Действительно, в момент удара мы считаем, что скорость мгновенно меняет свой знак. Быстрый, но непрерывный процесс изменения скорости во время удара мы заменяем разрывным процессом, и изменение скорости считаем мгновенным. Разрыв в данном случае — это следствие того, что мы пренебрегли непрерывными процессами деформации тел при ударе. Определенные трудности вызывает и отсутствие размеров у сталкивающихся точек.

Реальный удар двух тел заданной геометрической формы, во-первых, происходит в течение малого, но конечного времени, а во-вторых, приводит к деформации тел, что требует перехода к другой математической модели. Классическая механика предлагает свой способ преодоления этих трудностей, строит свою теорию удара, что позволяет решать задачи динамики с учетом соударений, но при этом все равно от классических динамических систем приходится переходить к гибридным.

В механике под ударом понимается действие бесконечно большой силы в течение короткого промежутка времени. Для описания явлений, происходящих при ударе, вводится понятие конечного ударного импульса $S = \lim\limits_{\substack{\tau \to 0 \\ F \to \infty}} \int\limits_0^\tau F(t)dt < \infty$ для не мгновенной силы F. Предполагается, что под действием бесконечно большой мгновенной силы, положение точки ее приложения к телу с массой m не меняется, а скорость претерпевает скачок $v^+ = v^- + \frac{S}{m}$. Именно такие процессы и описывает примитивный гибридный автомат, у которого в результате смены поведения меняются скачком начальные условия интегрируемой системы уравнений.

Удары делятся на центральные и не центральные (нецентральные). При центральном ударе линия соприкосновения двух тел лежит на прямой, соединяющих их центры, а касательная плоскость, проведенная в точке соприкосновения к ним, перпендикулярна к этой линии. Если при этом скорости центров тяжестей тел до удара лежат на линии центров, то удар называется прямым, в противном случае — косым. В зависимости от восстановления формы соударяющихся тел после удара (не сохраняется, сохраняется частично, сохраняется полностью) различают абсолютно неупругие, упругие и абсолютно упругие удары.

При абсолютно неупругом центральном ударе два тела с массами m_1 и m_2, двигавшиеся со скоростями v_1^-, v_2^- до удара, начинают двигаться по линии центров как единое тело со скоростью $v = v_1^+ = v_2^+ = \dfrac{m_1 v_1^- + m_2 v_2^-}{m_1 + m_2}$ после удара.

Упругий удар рассматривается как последовательность абсолютно неупругого удара, в результате которого тела слипаются, на мгновение превращаются в единое тело, после чего, в зависимости от коэффициента восстановления $0 \le k \le 1$, продолжают движение с новыми скоростями. Для прямого центрального удара значения новых скоростей вычисляется по формулам:

$$\begin{aligned} v &= \frac{m_1 v_1^- + m_2 v_2^-}{m_1 + m_2}, \\ v_1^+ &= v + k \cdot (v - v_1^-), \\ v_2^+ &= v + k \cdot (v - v_2^-). \end{aligned} \qquad (3.36)$$

Формулы (3.36) не вытекают из законов динамики — они следуют из законов сохранения импульса и энергии на всех стадиях столкновения.

В справедливости этого подхода можно убедиться на еще более простом примере. Путь первый шар ударяется о покоящийся второй

шар. Закон сохранения импульса и энергии запишем в виде системы двух уравнений:

$$\begin{cases} m_1 v_1^- = m_1 v_1^+ + m_2 v_2^+ \\ m_1 \dfrac{\left(v_1^-\right)^2}{2} = m_1 \dfrac{\left(v_1^+\right)^2}{2} + m_2 \dfrac{\left(v_1^+\right)^2}{2} \end{cases},$$

одно из решений которого, «физическое»:

$$v_1^+ = \frac{m_1 - m_2}{m_1 + m_2} v_1^-; \qquad v_1^+ = \frac{2m_1}{m_1 + m_2} v_1^-$$

очевидно, совпадает с формулами (3.36).

Таким образом, теория удара в классической механике — это пример использования гибридных систем, применяемых для моделирования уже не одним поколением физиков.

Для нас же важно, что в гибридных системах для определения новых начальных условий в момент смены поведения приходится использовать дополнительную информацию, позволяющую построить систему уравнений, связывающих значения переменных состояния во временной щели. Дополнительная информация нужна и для выбора одного решения этой системы, если оказывается, что их несколько.

В нашей системе двух материальных точек все эти сложные проблемы «скрыты» в операторе $V_x := -V_x$, который появляется в результате решения уравнений, возникающих при столкновении материальной точки с плоскостью бесконечно большой массы.

4. МНОГОКОМПОНЕНТНЫЕ СИСТЕМЫ С «КОНТАКТАМИ–ПОТОКАМИ»

Задания этого раздела используются при изучении методов построения моделей из компонентов с внешними переменными типа «контакты–потоки».

Как хорошо известно, электрические, механические, гидравлические и другие типы моделей, основанные на компонентных (аналогичных уравнениям для R, L, C компонентов электрических схем) и

топологических уравнениях (для электрических схем это уравнения Кирхгофа), могут рассматриваться и изучаться как аналогичные объекты различной физической природы, представленные единой математической моделью. В нашем случае в качестве базовых физических объектов выбраны электрические цепи.

Используемые электрические цепи настолько просты, что мы рекомендуем предварительно построить итоговую систему уравнений для электрической цепи вручную и сравнить ее с системой, построенной пакетом Rand Model Designer. Анализ системы, построенной автоматически, позволяет легко обнаружить исходные и компонентные, и топологические законы. Это помогает лучше понять материал раздела «Машинные методы формирования итоговых систем уравнений для моделей с компонентами с внешними переменными типа «вход–выход» и «контакт–поток».

ЗАДАНИЕ 1

Для указанной электрической схемы (см. рис. 4.1) разработать устройство, состоящее из двух блоков.

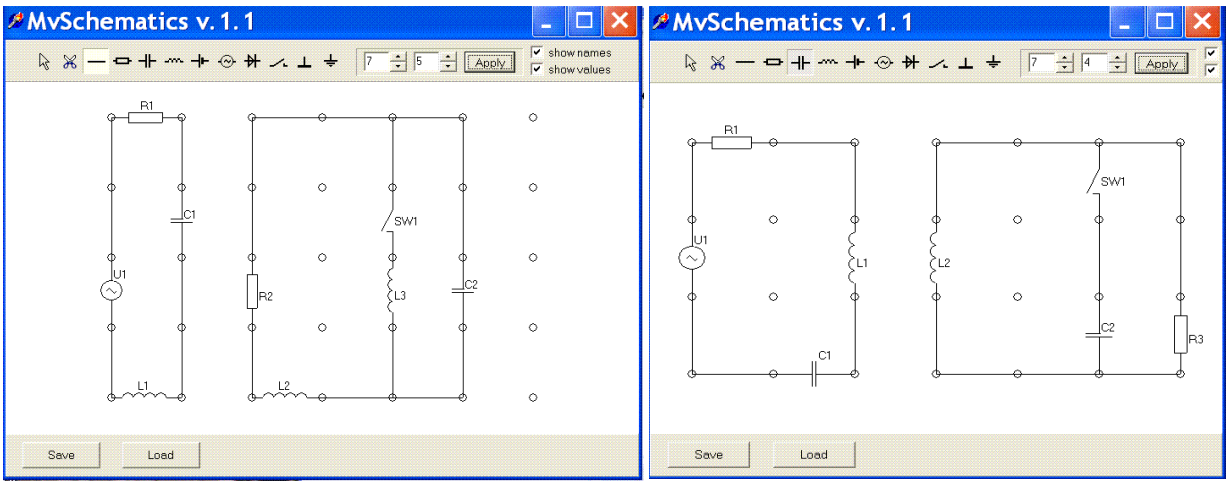


Схема 1 Схема 2

Рис. 4.1. Схемы к заданию 1

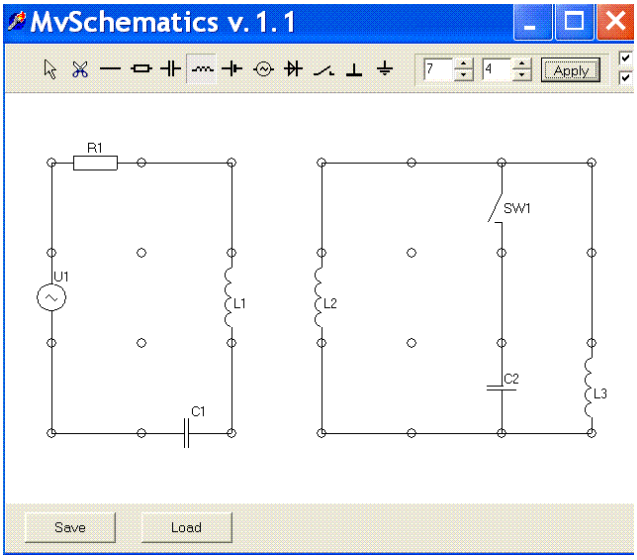


Схема 3

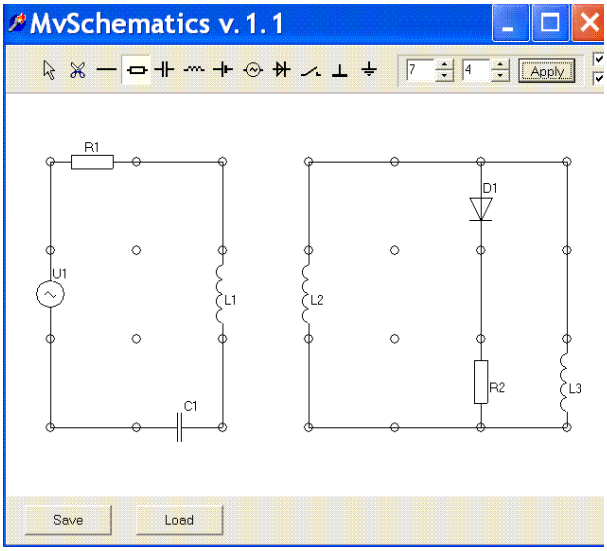


Схема 4

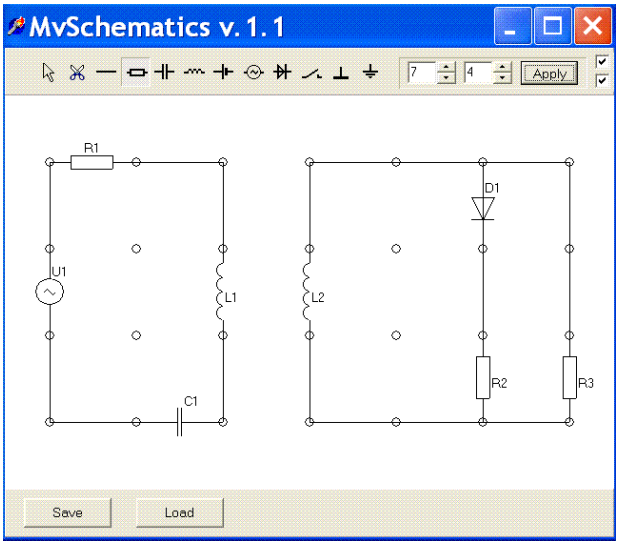


Схема 5

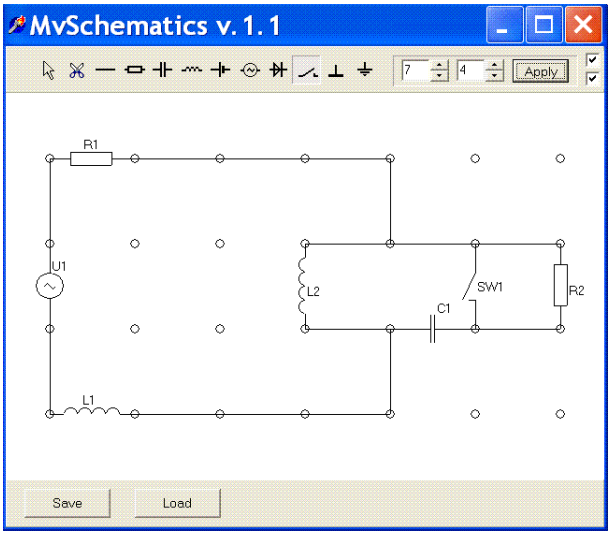


Схема 6

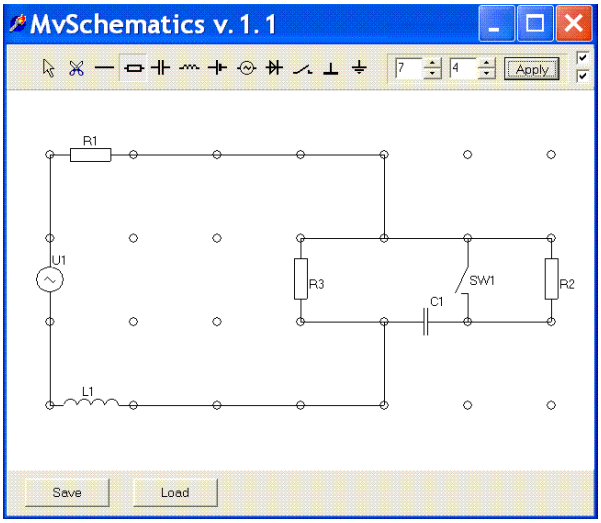


Схема 7

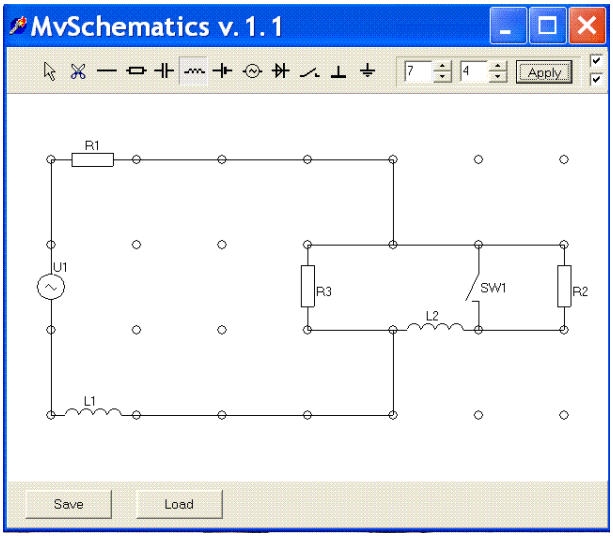


Схема 8

Рис. 4.1. Схемы к заданию 1 (продолжение)

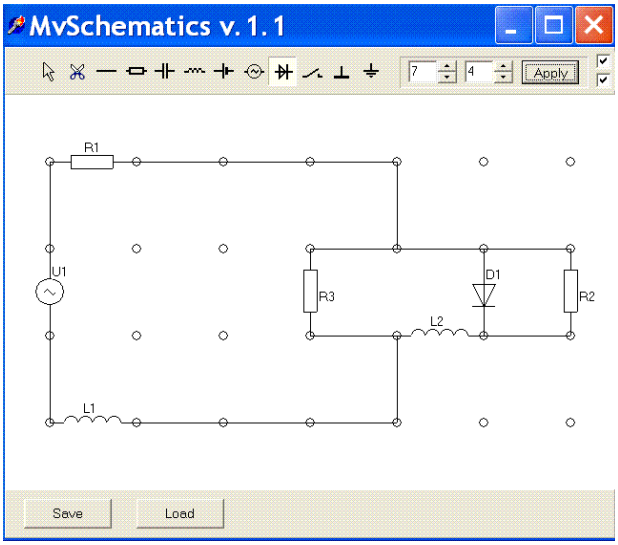


Схема 9

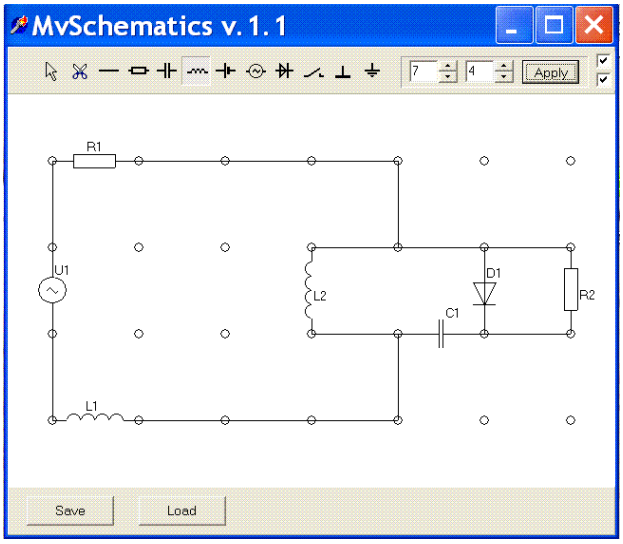


Схема 10

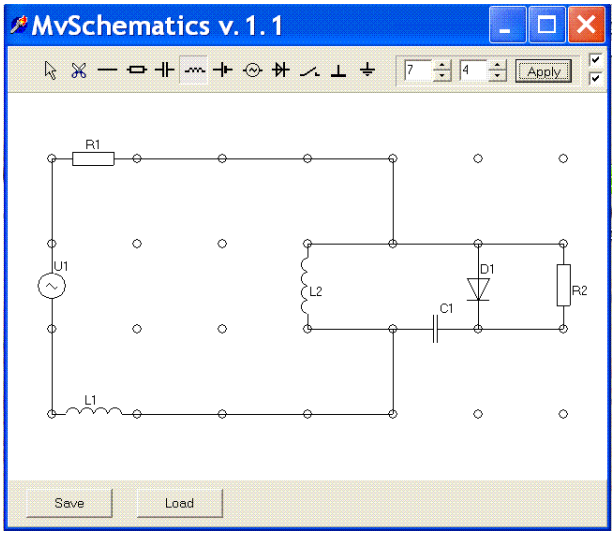


Схема 11

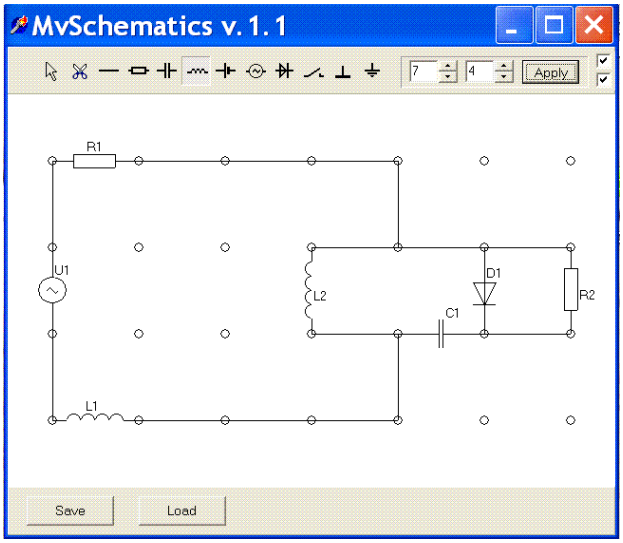


Схема 12

Рис. 4.1. Схемы к заданию 1 (окончание)

В схемах 1 – 6 такими устройствами должны быть два контура, связанные взаимной индуктивностью M.

В схемах 7 – 12 в качестве блоков выбирают внешний контур, содержащий внутренний конур, и сам внутренний контур.

Значения параметров выбрать так, чтобы в одном или обоих контурах наблюдались колебания.

В качестве проверки сравнить поведение многокомпонентного устройства с однокомпонентным устройством, содержащим уравнения всей схемы без искусственного разделения на блоки.

ЗАДАНИЕ 2

Исследуйте в RMD следующие цепи (см. рис. 4.2).

Постройте уравнения этих цепей вручную. Обратите внимание, что на элементы цепи напряжение может быть подано либо с правого источника, либо с левого, либо одновременно. Рассмотрите последний случай отдельно. Какие особенности построения уравнений возникают в этом случае?

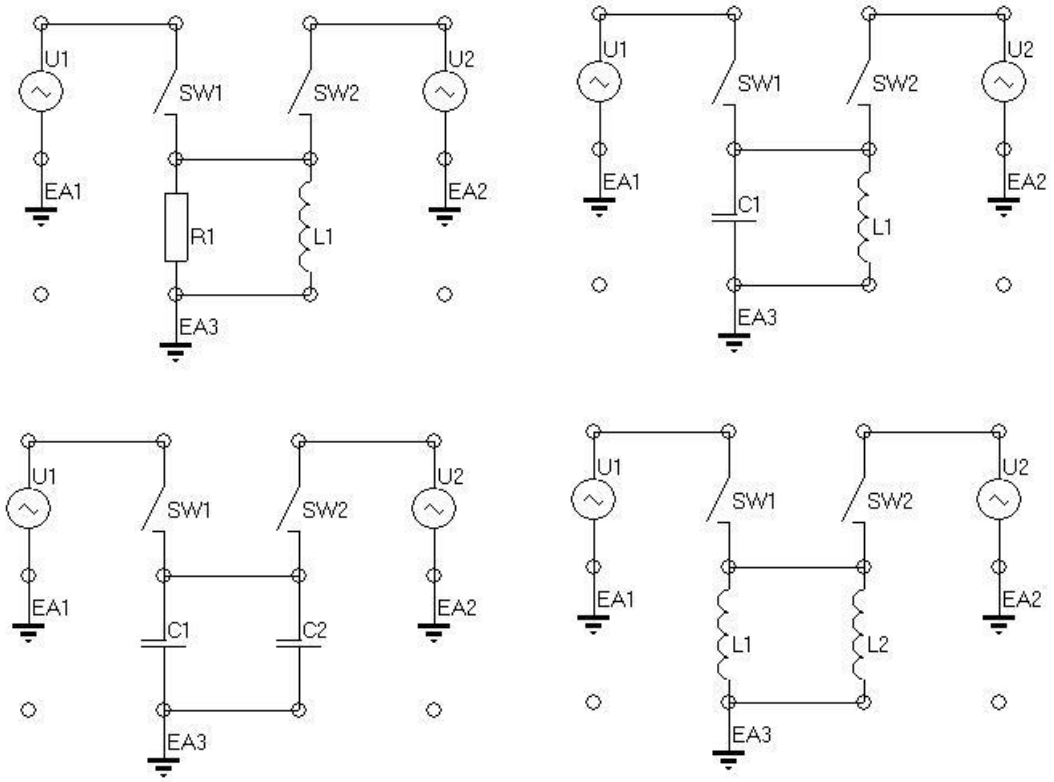


Рис. 4.2. Схемы к заданию 2

ЗАДАНИЕ 3

Постройте в RMD модели цепей, приведенных на рис. 4.3. Постройте гибридный автомат, реализующий все возможности подачи напряжения на элементы. Какие особенности возникают, при работе двух источников одновременно.

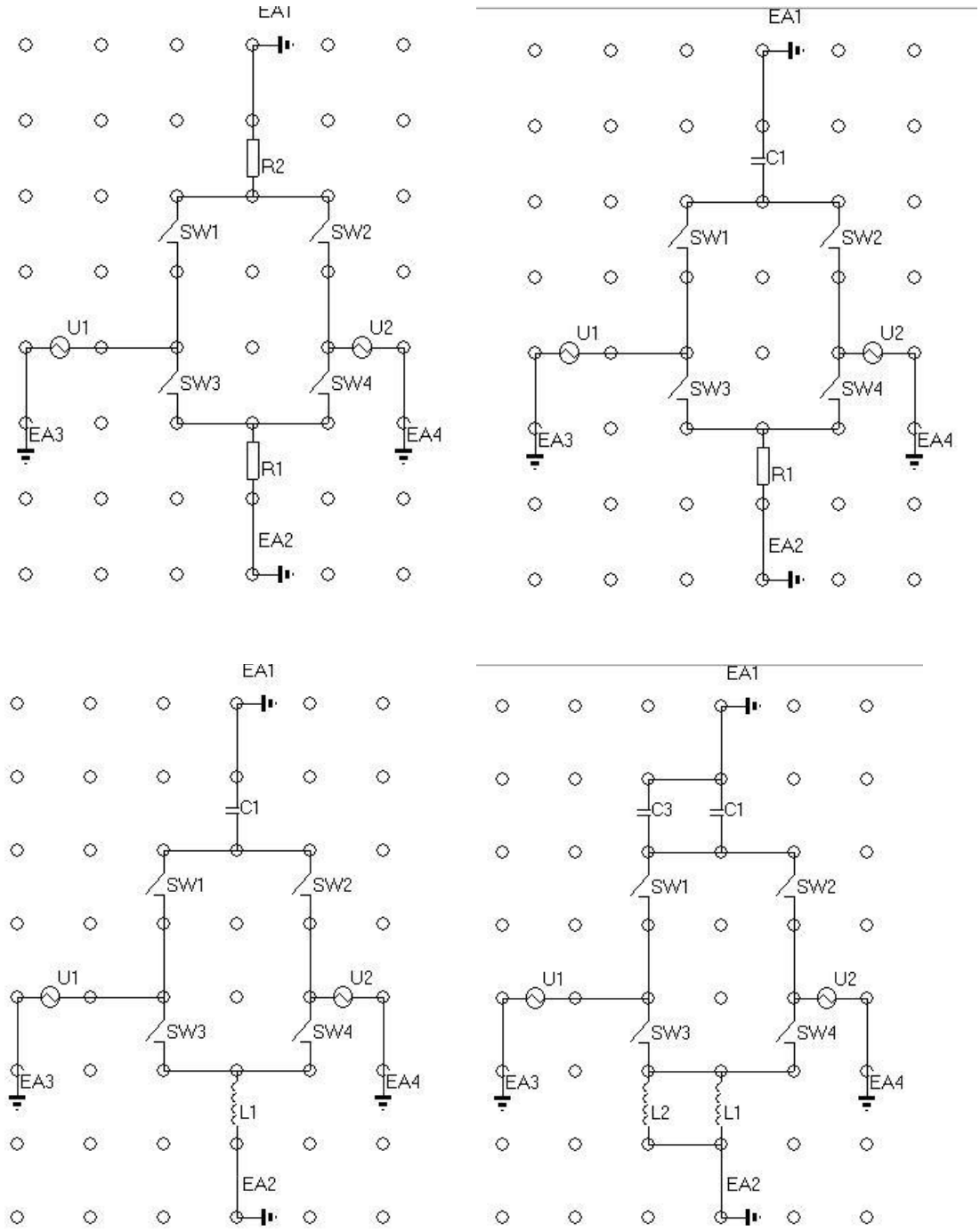


Рис. 4.3. Схемы к заданию 3

ЗАДАНИЕ 4

В этом задании один из ключей «электронный». Постройте гибридный автомат, соответствующий схемам, приведенным на рис. 4.4. Рассмотрите все возможные случаи подачи напряжения на элементы цепи.

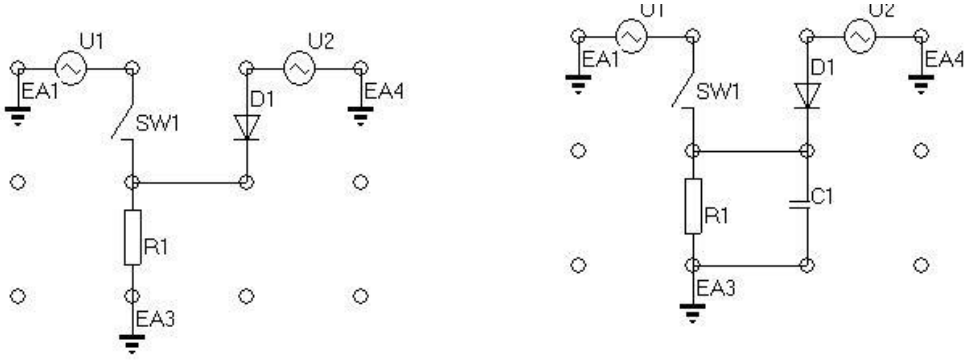


Рис. 4.4. Схемы к заданию 4

ЗАДАНИЕ 5

В книге [4] приведена табл. 4.1. Эта таблица говорит об аналогии между электрическими и механическими системами. Постройте мини-библиотеки для «Механической поступательной системы» (табл. 4.2) и «Механической вращательной системы» (табл. 4.3).

Таблица 4.1

Динамические аналогии

	Электрическая система	Механическая поступательная система	Механическая вращательная система
R	$R = \frac{e}{i}$	$R_m = \frac{f_m}{v}$	$R_R = \frac{f_R}{v}$
L	$e = L\frac{di}{dt}$	$f_m = m\frac{dv}{dt}$	$f_R = I\frac{dv}{dt}$
C	$i = L\frac{de}{dt}$	$f_m = \frac{x}{C_m}$	$f_R = \frac{\varphi}{C_R}$

Таблица 4.2

Механическая поступательная система

R_m	механическое поступательное сопротивление
f_m	приложенная механическая сила
x	перемещение
v	скорость
m	масса
C_m	эластичность упругого поступательного элемента

Таблица 4.3

Механическая вращательная система

R_R	механическое поступательное сопротивление
f_R	приложенная механическая сила
φ	угловое перемещение
v	угловая скорость
I	Момент инерции
C_R	эластичность упруго вращательного элемента

ЗАДАНИЕ 6

На рис. 4.5 приведены электрические схемы. Постройте их аналоги для «Механической поступательной системы» и «Механической вращательной системы».

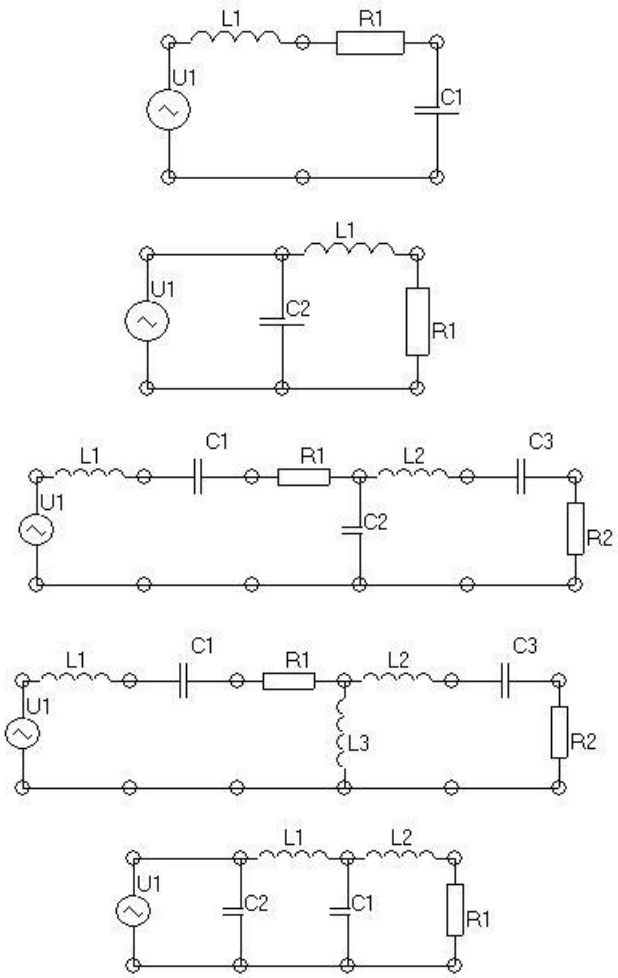


Рис. 4.5. Схемы к заданию 6

ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Электрические цепи, знакомые нам со школьных лет, когда мы изучали закон Ома — прекрасный пример для обсуждения возможностей и особенностей компонентного моделирования.

В середине прошлого века сопротивления, конденсаторы, индуктивности изготовлялись в виде кубиков, цилиндров или в форме капельки, раскрашивались в яркие цвета по принадлежности (все сопротивления одного наминала, например, были ярко красными) и имели две легко гнущиеся проволочки-контакты. Производство новых электронных устройств предусматривало разработку опытного образца, его испытание, создание технологии массового производства и выпуск нужного количества устройств. При изготовлении опытного образца идеальная модель — схема, нарисованная на бумаге — превращалась в реальное (физическое) устройство. Плату размечали и в нужных местах сверлили отверстия для металлических лепестков-контактов, в отверстиях закрепляли лепестки и припаивали к ним нужные детали, предварительно рассортированные и уложенные в отдельные коробочки. Совершенство технологии восхищало — стандартизованная элементная база, автоматизированная процедура разметки, конвейерная схема сборки — это ли не демонстрация красоты инженерной мысли!

Попробуем воспроизвести технологию моделирования и испытания простейших электронных схем на вычислительной машине с помощью пакета RMD.

Рассмотрим схемы, состоящие из идеальных источников тока и напряжения и линейных моделей сопротивления, индуктивности и емкости.

Напомним, что идеальным источником постоянного напряжения (U) называется компонент, поддерживающий заданное значение напряжения на постоянном уровне при любом протекающем токе (I). Аналогично, идеальным источником постоянного тока (U) называется

компонент, поддерживающий заданное значение тока на постоянном уровне при любом напряжении (U).

$$\forall U, I = I_0, \tag{4.1}$$

$$\forall I, U = U_0. \tag{4.2}$$

Идеальным источником переменного напряжения (тока), можно назвать компонент, реализующий заданную функцию U =U(t) (I = I(t)) от времени, вне зависимости от протекающего через него тока (поданного на него напряжения). Например,

$$\forall U, I = I_0 \cdot \sin(\omega \cdot t), \tag{4.3}$$

$$\forall I, U = U_0 \cdot \sin(\omega \cdot t). \tag{4.4}$$

Сопротивлением назовем компонент (линейный в соответствии с законом Ома), удовлетворяющий уравнению U=r*I. Компонентное уравнение содержит две переменные. Какая из них искомая, для какого значения второй переменной ее рассчитывать?

Попробуем ответить на вопрос. Рассмотрим компонентное уравнение *Сопротивления*

$$U = r \cdot I,$$

но перепишем его в виде

$$U - r \cdot I = 0. \tag{4.5}$$

Формально в написанном уравнении три переменных U, r, I, то есть оно не доопределено (в дальнейшем не доопределенная система называется недоопределенной). Искомых переменных две — U, I, так как r — явно параметр, со значением по умолчанию, например, $r = 1, r \in \Re^1$. Переменных две, уравнение одно. Неопределенность исчезает, только если мы указываем, в какой схеме используется *Сопротивление*.

- В электрической схеме, состоящей из *Сопротивления* и *Источника Тока*, появляется нужное нам второе уравнение

$$\begin{cases} U - r \cdot I = 0 \\ I - I_0 \cdot \sin(\omega \cdot t) = 0 \end{cases}, \tag{4.6}$$

или последовательность формул (вычислимая последовательность формул)

$$\begin{cases} I = I_0 \cdot \sin(\omega \cdot t) \\ U = r \cdot I = 0 \end{cases}. \tag{4.7}$$

• В электрической схеме, состоящей из *Сопротивления* и *Источника Напряжения,* имеем:

$$\begin{cases} U = U_0 \cdot \sin(\omega \cdot t) \\ I = \dfrac{U}{r} \end{cases}. \tag{4.8}$$

Вопросы для самоконтроля

1. Рассмотрим устройства *Источник Постоянного Тока* и *Источник Постоянного Напряжения*. Если к ним подключить измерительный прибор, то мы увидим определяемые уравнениями источника значения тока или напряжения. Какое уравнение или система уравнений возникнет при подключенном к источнику измерительном приборе? Напишите уравнения измерительных приборов («Вольтметра» и «Амперметра») и решите их.

2. Компонентные уравнения *Сопротивления* недоопределены. В то же время, если мы попытаемся измерить ток или напряжение отдельно взятого *Сопротивления,* то прибор покажет нулевые значения. Можно ли при подключенном измерительном приборе говорить о недоопределенной системе для *Сопротивления*? Нарисуйте схему и напишите систему уравнений, возникающую при подключении к сопротивлению «Вольтметра» или «Амперметра».

Дополним нашу библиотеку компонентами *Емкость* и *Индуктивность*

$$C \cdot \frac{dU}{dt} - I = 0, \tag{4.9}$$

$$U - L \cdot \frac{dI}{dt} = 0. \tag{4.10}$$

Рассмотрим цепи, состоящие из *Емкости и Источников питания*, и *Индуктивности и Источников питания*.

Перечень простейших электрических цепей приведен в табл. 4.4.

Таблица 4.4

Простейшие электрические цепи для *Сопротивления*, *Емкости* и *Индуктивности*

№	Название	Система уравнений цепи
1	Источник постоянного тока и Сопротивление	$\begin{cases} I = I_0 \\ U - r \cdot I = 0 \end{cases}$
2	Источник постоянного напряжения и Сопротивление	$\begin{cases} U = U_0 \\ U - r \cdot I = 0 \end{cases}$
3	Источник постоянного тока и Индуктивность	$\begin{cases} \mathrm{I} = \mathrm{I}_0 \\ U - L \cdot \dfrac{dI}{dt} = 0 \end{cases}$
4	Источник постоянного напряжения и Емкость	$\begin{cases} \mathrm{U} = \mathrm{U}_0 \\ C \cdot \dfrac{dU}{dt} - I = 0 \end{cases}$
5	Источник синусоидального тока и Индуктивность	$\begin{cases} \mathrm{I} = \mathrm{I}_0 \cdot \sin(\omega \cdot t) \\ U - L \cdot \dfrac{dI}{dt} = 0 \end{cases}$
6	Источник синусоидального напряжения и Емкость	$\begin{cases} \mathrm{U} = \mathrm{U}_0 \cdot \sin(\omega \cdot t) \\ C \cdot \dfrac{dU}{dt} - I = 0 \end{cases}$
7	Источник синусоидального напряжения и Индуктивность	$\begin{cases} U = \mathrm{U}_0 \cdot \sin(\omega \cdot t) \\ U - L \cdot \dfrac{dI}{dt} = 0 \end{cases}$
8	Источник синусоидального тока и Емкость	$\begin{cases} \mathrm{I} = \mathrm{I}_0 \cdot \sin(\omega \cdot t) \\ C \cdot \dfrac{dU}{dt} - I = 0 \end{cases}$

Системы для цепей № 1 и 2 интуитивно понятны и легко решаемы — в каждой линейное алгебраическое уравнение, дополненное формулой.

В системах для цепей № 3 и 4, так как токи (напряжения) постоянны, можно отбросить дифференциальные уравнения и заменить их формулами

$$\begin{cases} \mathrm{I} = \mathrm{I}_0 \\ U = U_0 \end{cases},$$

дополнив уравнения компонентов начальными значениями токов и напряжений U_0 (I_0) для написанных дифференциальных уравнений.

Системы для цепей № 5 и 6 выглядят на первый взгляд странно — дифференциальное уравнение (№ 5), которое нужно было бы решать относительно напряжения U

$$\left\{ C \cdot \frac{dU}{dt} - I = 0, \right.$$

уже имеет предписанное решение

$$\mathrm{U} = \mathrm{U}_0 \cdot \sin(\omega \cdot t),$$

и это не дифференциальное уравнение относительно U, а формула для расчета тока I. Однако цепи № 7 и 8 показывают, что приходится решать и дифференциальные уравнения! Получается, что для каждой конкретной цепи нужно уметь автоматически приводить компонентные уравнения к форме, которая позволит найти решение (аналитически или численно).

Сведем информацию о компонентах библиотеки базовых электрических линейных элементов в единую таблицу (табл. 4.5).

Теперь поговорим об изображении элементов. Любой электрический элемент из нашей библиотеки можно изобразить прямоуголь-

ником с двумя выводами (полюсами «по-электрически») или двухполюсником (рис. 4.6).

Таблица 4.5

Электрические компоненты. Мини-библиотека

№	Название	Уравнение
1	Источник постоянного тока	$\forall U, I = I_0$
2	Источник постоянного напряжения	$\forall I, U = U_0$
3	Источник синусоидального тока	$\forall U, I = I_0 \cdot \sin(\omega \cdot t)$
4	Источник синусоидального напряжения	$\forall I, U = U_0 \cdot \sin(\omega \cdot t)$
5	Сопротивление	$U = r \cdot I$ $U(0) = 0; I(0) = 0$
6	Емкость	$C \cdot \frac{dU}{dt} = I$ $U(0) = U_0; I(0) = 0$
7	Индуктивность	$U = L \cdot \frac{dI}{dt}$ $U(0) = 0; I(0) = I_0$

Наш прообраз — прямоугольник с левым и правым выводами — похож на открытый компонент RMD с двумя внешними переменными. С программной точки зрения — это абстрактный класс, для которого достаточно определить типы внешних переменных.

При соединении таких компонентов возникает прообраз электрической цепи.

Рис. 4.6. Прообраз компонента из табл. 4.5

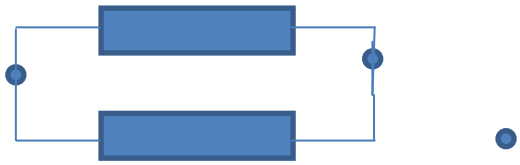


Рис. 4.7. Прообраз цепи

На рис. 4.7 изображены прообразы элементов и узлы соединения, образующие электрический контур. Левый и правый выводы однотипны. На левом и правом выводах можно измерить напряжение $U_{left} \equiv U_1; U_{right} \equiv U_2$, относительно некоторого «нулевого» узла с «нулевым» напряжением U_0 (этот узел расположен в правом нижнем углу рисунка). Будем считать, что ток втекает в элемент слева и вытекает справа — $I_{left} \equiv I_1; I_{right} \equiv I_2$. Таким образом, с каждым выводом связана пара переменных $((U_i; I_i); i = 1,2;)$. Эта пара представляется записями:

$$\begin{cases} \text{pin } Left\_Component : record(contactU : double; \text{flow } I : double) \\ \text{pin } Right\_Component : record(contactU : double; \text{flow } I : double) \end{cases}. \quad (4.11)$$

Если перевести это описание на язык RMD, то мы имеем непрерывный, открытый компонент элементарной структуры с двумя внешними переменными (выводами) — записями с двумя полями $(U_i; I_i)$ — или четырьмя скалярными переменными $(U_i; I_i); i = 1,2;$. Обе переменные записи вещественного типа, но разного вида. Напряжения принадлежат к виду «контакт», а токи — к виду «поток». При соединении («пайке») выводов в узлах выполняется второй закон Кирхгофа для токов (по определению потоков), и, соответственно, равны напряжения соединяемых выводов. С математической точки зрения при соединении возникают два дополнительных линейных алгебраических уравнения «связи» для каждого узла (верхние индексы — это номера соединяемых компонентов):

$$\begin{cases} U_2^1 - U_1^2 = 0 \\ I_2^1 + I_1^2 = 0 \end{cases}. \qquad (4.12)$$

$$Right\_Conponent\_1.U - Left\_Conponent\_2.U = 0,$$

$$Right\_Conponent\_1.I + Left\_Conponent\_2.I = 0.$$

Посмотрим, как связаны переменные $(U_i; I_i); i = 1,2;$ прототипа с переменными $(U; I)$ компонентов из табл. 4.6. Очевидно, что

$$U = U_2 - U_1,$$
$$I = I_1$$

или

$$U = U_1 - U_2,$$
$$I = I_2.$$

Для табличных элементов справедлив закон «сохранения» токов из (4.11), но он не выполняется автоматически. Его следует прописать явно и лучше это сделать в абстрактном классе — прототипе элемента, назовем его *Двухполюсником*. *Двухполюсник* имеет:

две внешние переменные:

$$\begin{cases} pin\ Left\_Two\_terminal\_circuit \\ pin\ Right\_Two\_terminal\_circuit \end{cases};$$

две внутренние:

$$u : double$$
$$i : double;$$

две формулы (разрешенные относительно искомых переменных уравнения):

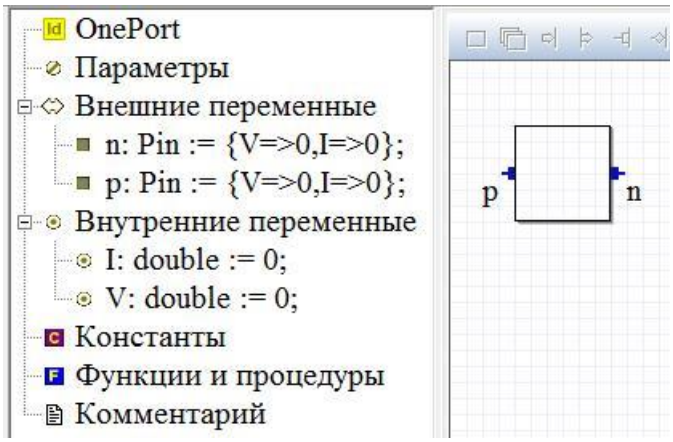
$$\begin{array}{l} u = Left\_Two\_terminal\_circuit.U - Right\_Two\_terminal\_circuit.U \\ i = Left\_Two\_terminal\_circuit.I \end{array};$$

и одно общее для всех рассматриваемых в этом примере электрических элементов уравнение

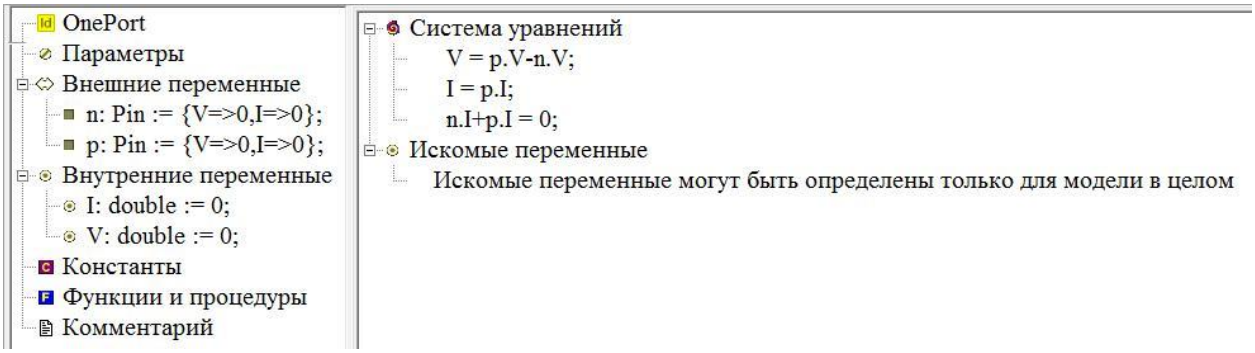
$$Left.I + Right.I = 0.$$

Этот класс чрезвычайно полезен (рис. 4.8), так все элементы из табл. 4.5 — потомки этого класса. Для получения *Сопротивления*,

Емкости, *Индуктивности* и *Источников питания* достаточно добавить соответствующие компонентные уравнения из табл. 4.5.

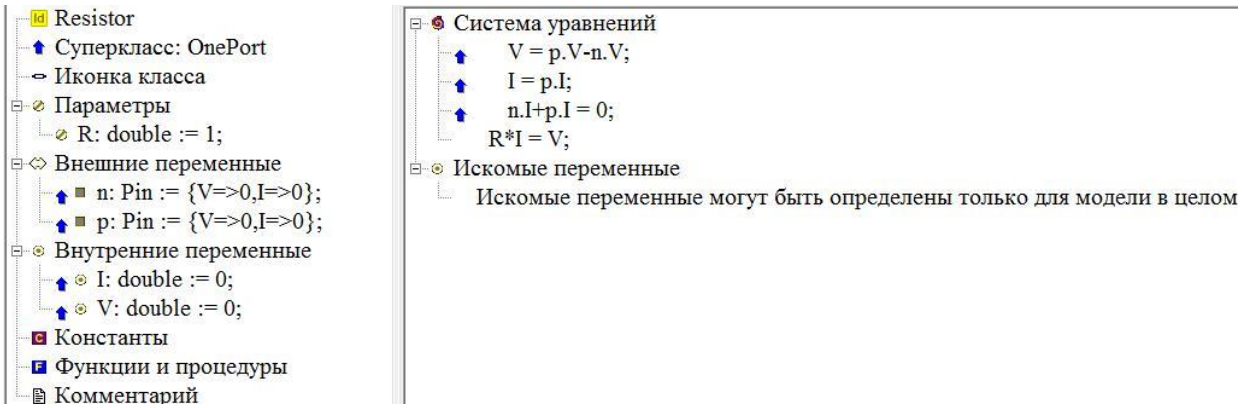


а) Интерфейс класса *Двухполюсник*



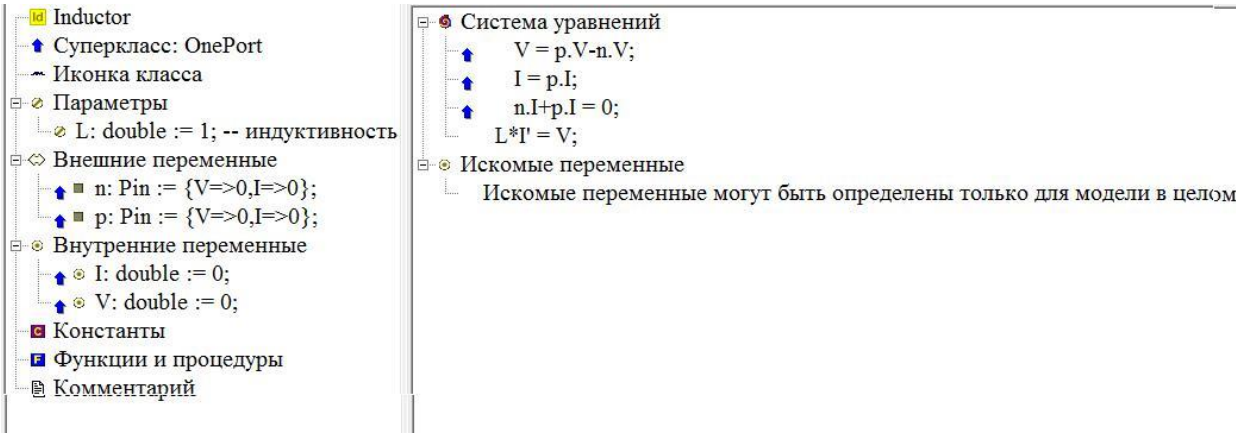
б) Уравнения класса *Двухполюсник*

Рис. 4.8. Класс *Двухполюсник*

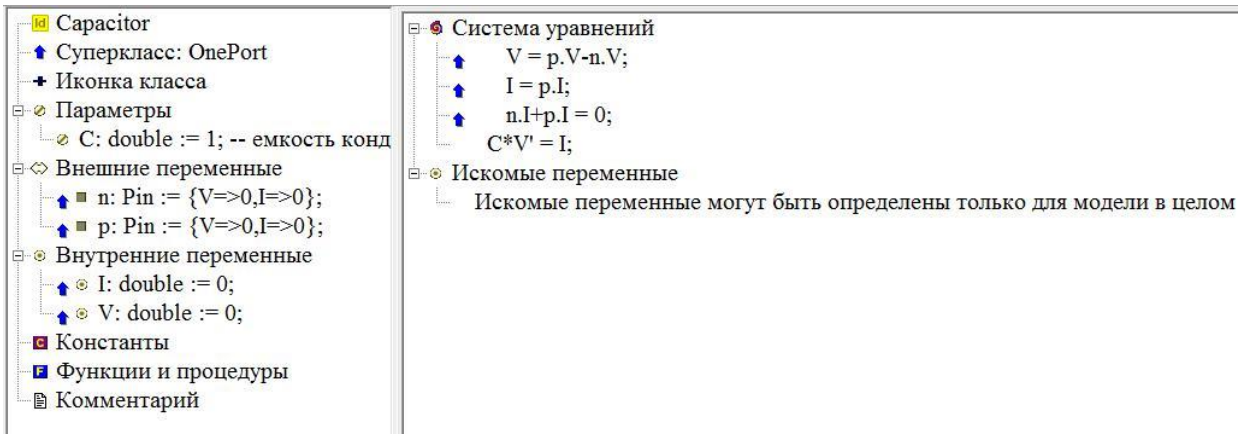


а) *Сопротивление*, уравнения

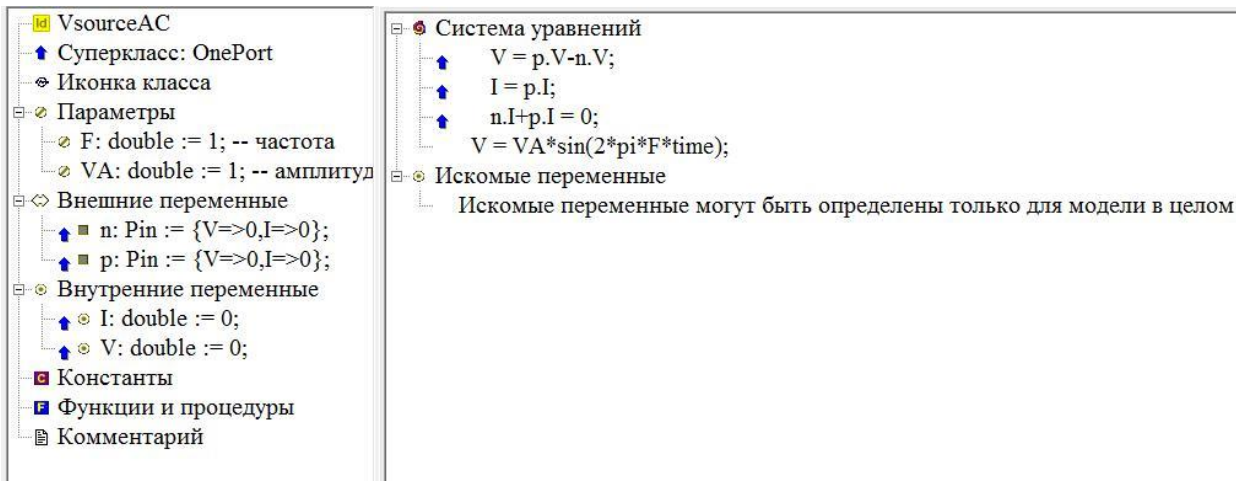
Рис. 4.9. Компоненты библиотеки «Электричество»



б) *Индуктивность*, уравнения



в) *Емкость*, уравнения



г) *Источник синусоидального напряжения*, уравнения

Рис. 4.9. Компоненты библиотеки «Электричество» (окончание)

Если реализовать описанные классы в проекте «Электричество» (рис. 4.9), то можно будет строить достаточно интересные для обучения электрические схемы (рис. 4.10).

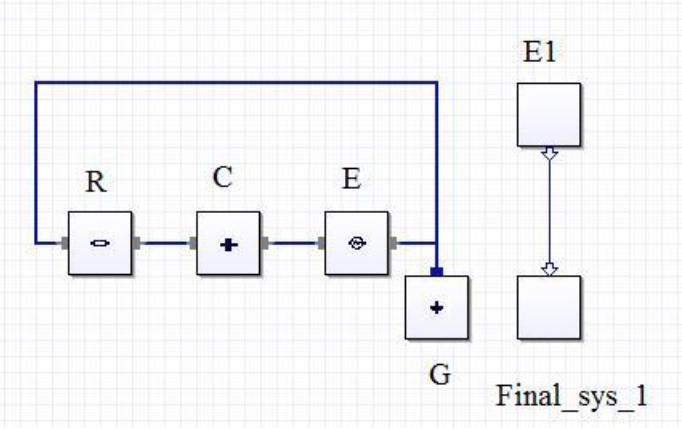


Рис. 4.10. RC-цепочка

Кратко рассмотрим, как формируются уравнения электрических цепей, которые можно создать при помощи нашей мини-библиотеки.

Рассмотрим цепь, показанную на рис. 4.10.

В нашей модели присутствует RC-цепочка (слева), на которую подано переменное напряжение, выполненная в стиле «физического моделирования», а рядом — ее аналог (справа), в виде компонентов с «входами–выходами». Вторая модель содержит уравнения, приведенные на рис. 4.11.

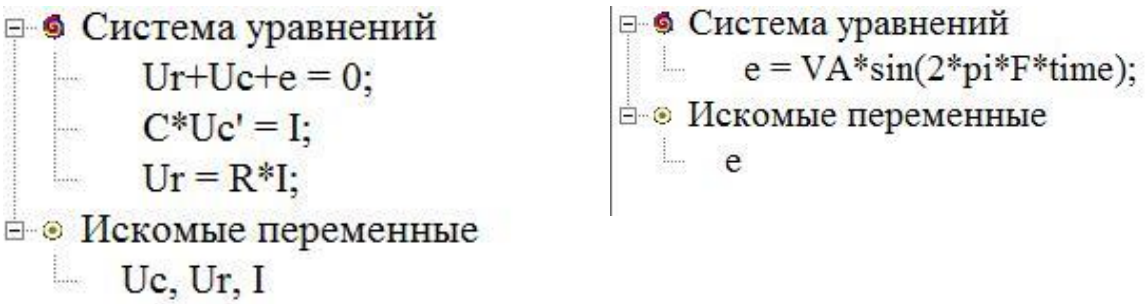


Рис. 4.11. Итоговая система, созданная вручную

Как видно из рис. 4.11, первое уравнение — это закон Кирхгофа, говорящий, что сумма падений напряжений при обходе контура равна нулю, остальные — компонентные уравнения для *Сопротивления* и *Емкости*. *Источник Напряжения* реализован как отдельный компонент.

На рис. 4.12 приведены результаты сравнения двух моделей.

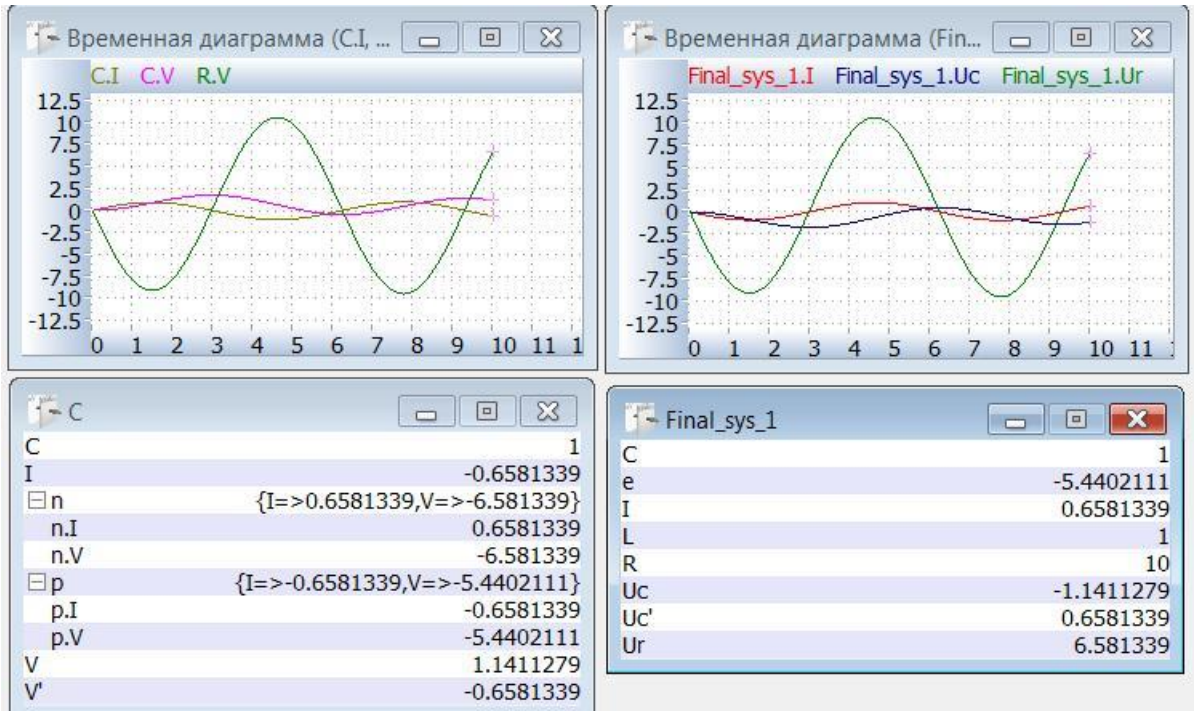
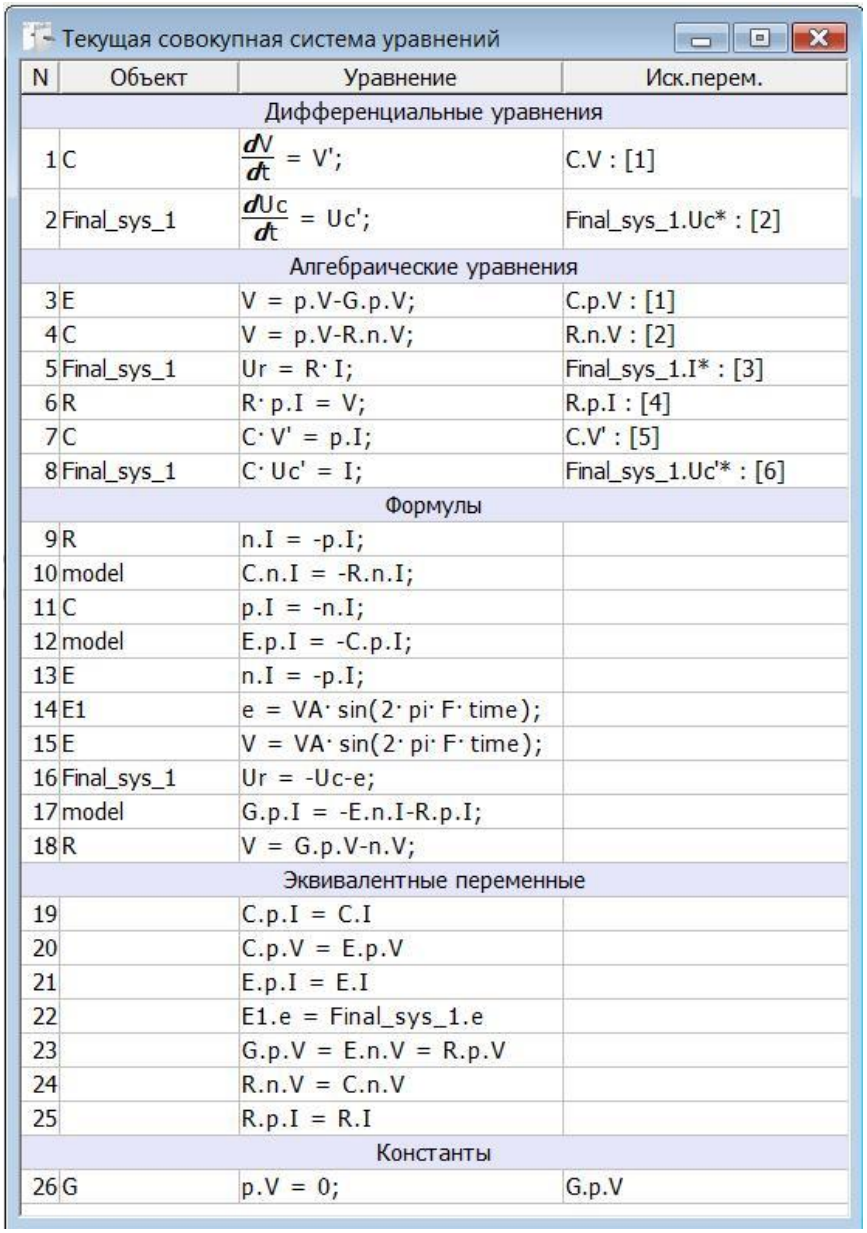


Рис. 4.12. Результаты сравнения

На рис. 4.13 приведена итоговая система, сформированная автоматически. Исключим из рассмотрения объекты _Fynal_sys_1 и E1, которые дают одно дифференциальное, два алгебраических уравнения и две формулы. На долю «физического моделирования приходится одно дифференциальное, четыре алгебраических уравнения и восемь формул. Что такое «эквивалентные» переменные проще всего пояснить на модели с «входами–выходами». Выходная переменная компоненты источника питания E1.e поступает на вход «Final sys_1.e» компонента RC-цепочка. Очевидно, что они равны, и достаточно одной из них.

Сначала посмотрим, во что превратилась уравнения эквивалентной системы — «Final_system_1». Дифференциальное уравнение, записанное в форме уравнения, не разрешенного относительно производной превратилось в систему алгебро-дифференциальных уравнений в форме Хессенберга. В таком виде с системой может справиться любой численный метод RMD. Из двух оставшихся линейных уравнений (закон Ома и закон Кирхгофа), закон Ома был признан уравнением, а закон Кирхгофа превратился в формулу.



Текущая совокупная система уравнений

N	Объект	Уравнение	Иск.перем.
		Дифференциальные уравнения	
1	C	$\frac{dV}{dt}$ = V';	C.V : [1]
2	Final_sys_1	$\frac{dUc}{dt}$ = Uc';	Final_sys_1.Uc* : [2]
		Алгебраические уравнения	
3	E	V = p.V-G.p.V;	C.p.V : [1]
4	C	V = p.V-R.n.V;	R.n.V : [2]
5	Final_sys_1	Ur = R· I;	Final_sys_1.I* : [3]
6	R	R· p.I = V;	R.p.I : [4]
7	C	C· V' = p.I;	C.V' : [5]
8	Final_sys_1	C· Uc' = I;	Final_sys_1.Uc'* : [6]
		Формулы	
9	R	n.I = -p.I;	
10	model	C.n.I = -R.n.I;	
11	C	p.I = -n.I;	
12	model	E.p.I = -C.p.I;	
13	E	n.I = -p.I;	
14	E1	e = VA· sin(2· pi· F· time);	
15	E	V = VA· sin(2· pi· F· time);	
16	Final_sys_1	Ur = -Uc-e;	
17	model	G.p.I = -E.n.I-R.p.I;	
18	R	V = G.p.V-n.V;	
		Эквивалентные переменные	
19		C.p.I = C.I	
20		C.p.V = E.p.V	
21		E.p.I = E.I	
22		E1.e = Final_sys_1.e	
23		G.p.V = E.n.V = R.p.V	
24		R.n.V = C.n.V	
25		R.p.I = R.I	
		Константы	
26	G	p.V = 0;	G.p.V

Рис. 4.13. Итоговая система, построенная автоматически

Нечто аналогичное произошло и с RC-цепочкой. Дифференциальное уравнение компоненты *Емкость* стало алгебро-дифференциальным. Остальные уравнения — компонентные уравнения.

Рассмотрим теперь цепочку, в которой на емкость подано напряжение (рис. 4.14). На рис. 4.14 три модели — «физическая» мо-

дель, эквивалентная ей изолированная система и система с «входами-выходами», в которой источник напряжения выполнен в виде отдельного компонента. В последней модели компонента Final_sys_Q_1 содержит три варианта эквивалентных систем, однако любая из них предполагает вычисление производной от напряжения, подаваемого на емкость. Это как раз тот самый случай, когда дифференциальное уравнение превращается в формулу. Модель Final_sys показывает (рис. 4.15, рис. 4.16), что можно использовать численное дифференцирование (e_prim_n), а можно вычислить производную вручную.

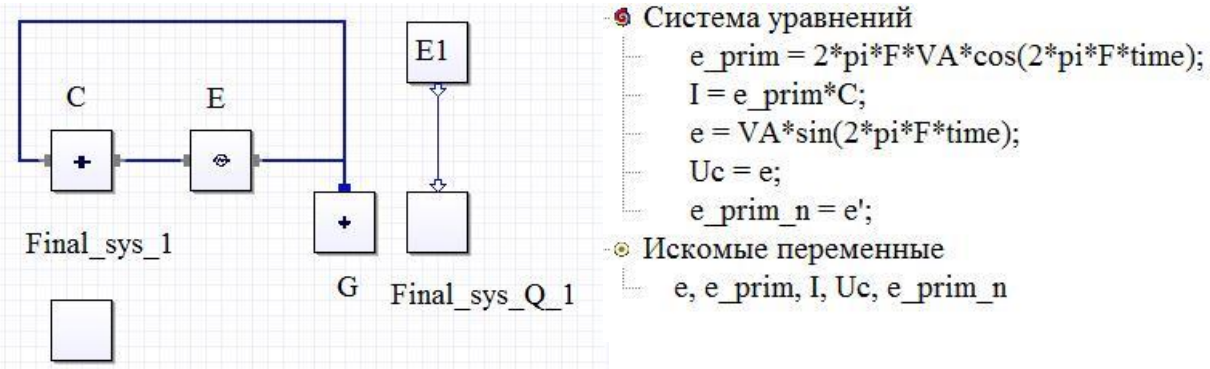


Рис. 4.14. Цепочка и ее эквивалентная система

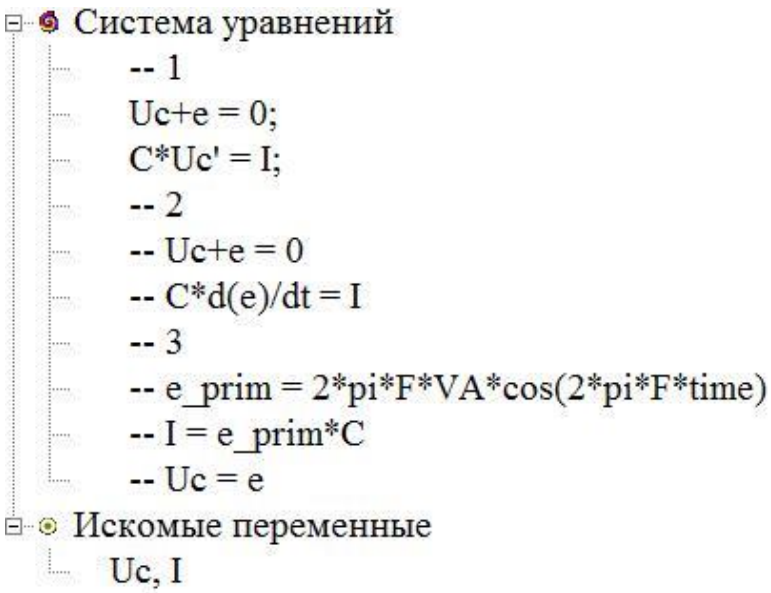
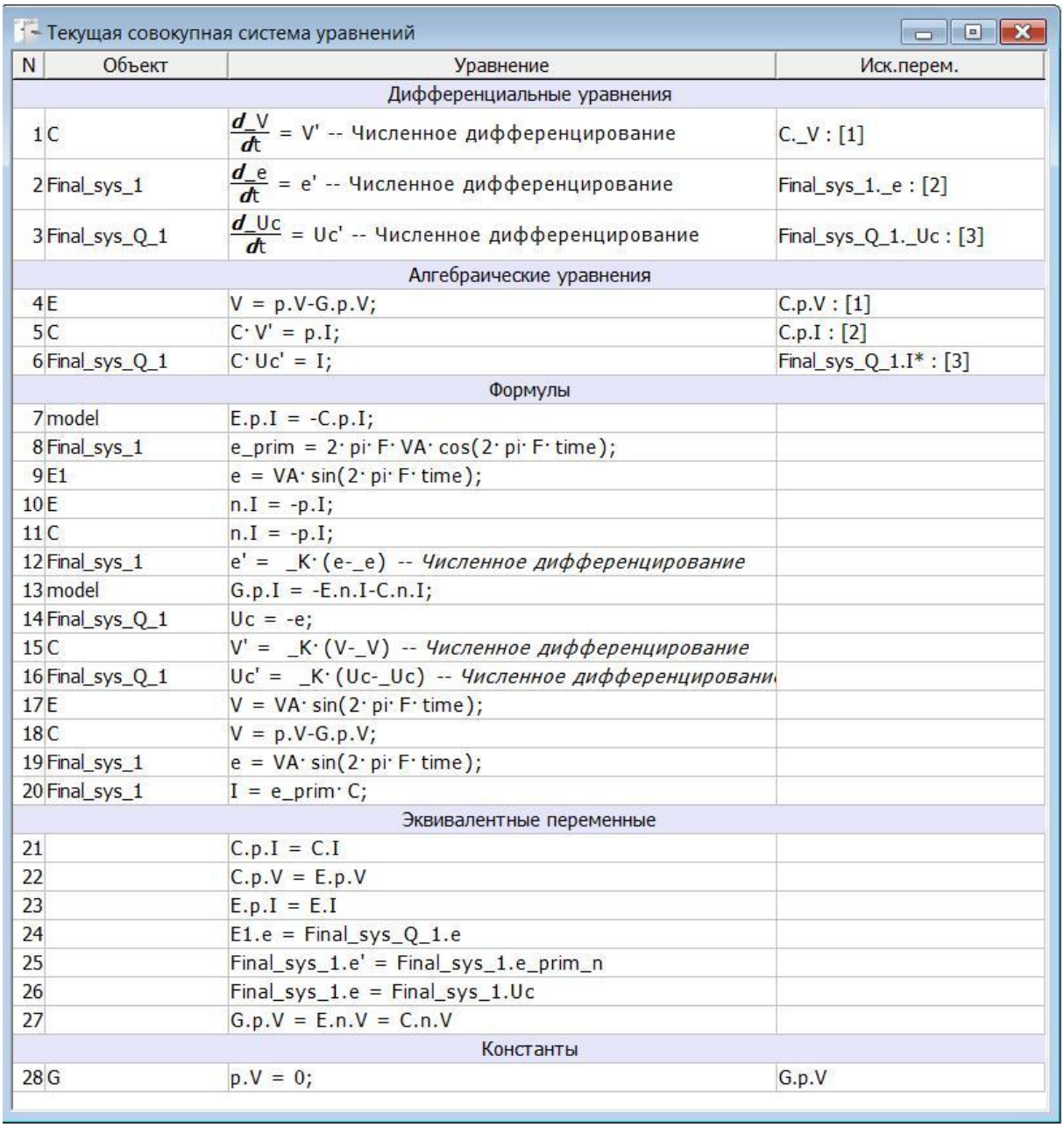


Рис. 4.15. Возможные формы эквивалентной системы для компонента Finalsys_Q_1



Текущая совокупная система уравнений

N	Объект	Уравнение	Иск.перем.
		Дифференциальные уравнения	
1	C	$\frac{d\_V}{dt}$ = V' -- Численное дифференцирование	C._V : [1]
2	Final_sys_1	$\frac{d\_e}{dt}$ = e' -- Численное дифференцирование	Final_sys_1._e : [2]
3	Final_sys_Q_1	$\frac{d\_Uc}{dt}$ = Uc' -- Численное дифференцирование	Final_sys_Q_1._Uc : [3]
		Алгебраические уравнения	
4	E	V = p.V-G.p.V;	C.p.V : [1]
5	C	C· V' = p.I;	C.p.I : [2]
6	Final_sys_Q_1	C· Uc' = I;	Final_sys_Q_1.I* : [3]
		Формулы	
7	model	E.p.I = -C.p.I;	
8	Final_sys_1	e_prim = 2· pi· F· VA· cos(2· pi· F· time);	
9	E1	e = VA· sin(2· pi· F· time);	
10	E	n.I = -p.I;	
11	C	n.I = -p.I;	
12	Final_sys_1	e' = _K· (e-_e) -- *Численное дифференцирование*	
13	model	G.p.I = -E.n.I-C.n.I;	
14	Final_sys_Q_1	Uc = -e;	
15	C	V' = _K· (V-_V) -- *Численное дифференцирование*	
16	Final_sys_Q_1	Uc' = _K· (Uc-_Uc) -- *Численное дифференцировани*	
17	E	V = VA· sin(2· pi· F· time);	
18	C	V = p.V-G.p.V;	
19	Final_sys_1	e = VA· sin(2· pi· F· time);	
20	Final_sys_1	I = e_prim· C;	
		Эквивалентные переменные	
21		C.p.I = C.I	
22		C.p.V = E.p.V	
23		E.p.I = E.I	
24		E1.e = Final_sys_Q_1.e	
25		Final_sys_1.e' = Final_sys_1.e_prim_n	
26		Final_sys_1.e = Final_sys_1.Uc	
27		G.p.V = E.n.V = C.n.V	
		Константы	
28	G	p.V = 0;	G.p.V

Рис. 4.16. Автоматически сформированная система для цепочки, состоящей из *Емкости* и *Источника Напряжения*

ПРИЛОЖЕНИЯ

Приложение 1

ВВЕДЕНИЕ В ПАКЕТ MATLAB

Пакет **MATLAB** [5,6] родился в результате стремления вычислителей расширить сферу применения хороших численных методов за счет создания простого и выразительного языка, позволяющего пользователю указать вычислительной машине, *что* нужно решать, в отличие от традиционных языков, где требуется подробно указать машине *как* нужно решать.

Популярные среди профессионалов специализированные коллекции **LINPACK** (решение систем линейных алгебраических уравнений) и **EISPACK** (решение проблемы собственных значений) достаточно сложны для использования, т. к. необходимо, с одной стороны, хорошо знать Фортран, на котором они написаны, а с другой — реализованные в них численные методы и структуру библиотек **LINPACK** и **EISPACK.**

Потребность упростить использование этих специализированных коллекций в практических вычислениях и привела к созданию пакета **MATLAB.** Включенные в язык пакета операторы, реализующие основные возможности коллекций **LINPACK**, **EISPACK** (найти собственные числа, решить систему уравнений и т. д.), дали возможность пользователям, далеким от вычислительной математики, легко и эффективно использовать лучшие программные реализации численных методов, практически ничего не зная о том, как они устроены.

Современный **MATLAB** — это вычислительный пакет, предназначенный для решения задач, в основе которых лежат матричные вычисления. Основная форма работы пакета — диалоговая. Пакет содержит специальные библиотеки для решения многих практически важных задач. Существует возможность создавать свои собственные библиотеки программ и, тем самым, расширять возможности пакета. Пакет оснащен 2D- и 3D-графикой.

Структура пакета MATLAB

Структура пакета представлена на рис. П1.1.

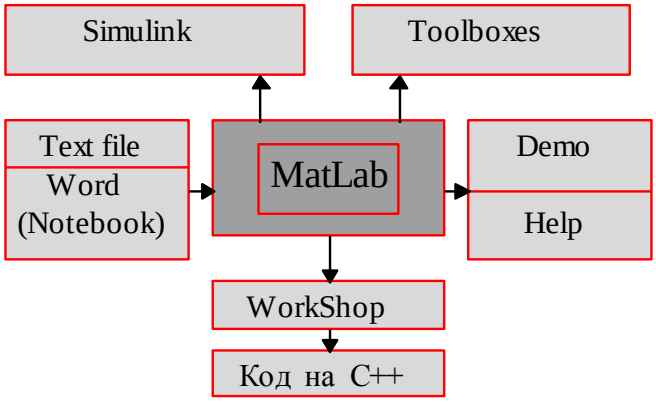


Рис. П1.1. Структура пакета MATLAB

Исходные данные и программы на языке пакета **MATLAB** можно вводить непосредственно с клавиатуры или подготавливать их в виде текстовых файлов. Текстовые файлы с программами должны обязательно иметь расширение .m — "имя.m". Дополнительные возможности ввода предоставляет программа **Notebook**. Она позволяет общаться с пакетом **MATLAB**, используя текстовый редактор **Word.** При таком способе и сами команды, и результат их выполнения размещаются в одном окне редактора **Word**. Используя возможности редактора, зафиксированный таким образом протокол работы можно легко превратить в отчет или научную статью.

Пакет **MATLAB** поставляется вместе с обширным набором специализированных библиотек (**Toolboxes**) для решения практически важных задач. На его базе разработана программа **Simulink**, предназначенная для моделирования сложных динамических систем. Традиционная для всех пакетов демонстрационная программа **Demo** является одновременно электронным учебником. Включенные в нее примеры сопровождаются подробными пояснениями. Она может быть использована для самостоятельного первоначального изучения входно-

го языка пакета. С помощью приложения **WorkShop** можно перевести программы, написанные на языке пакета **Matlab**, в код на C++.

Вызов пакета MATLAB

Для начала работы с пакетом **MATLAB** достаточно выбрать в Главном меню (кнопка «Start» в нижнем левом углу экрана, если вы работаете с операционной системой Windows) команду «Programs/MATLAB for Windows/MATLAB», и вы увидите основное окно пакета «MATLAB Command Window» (рис. П1.2).

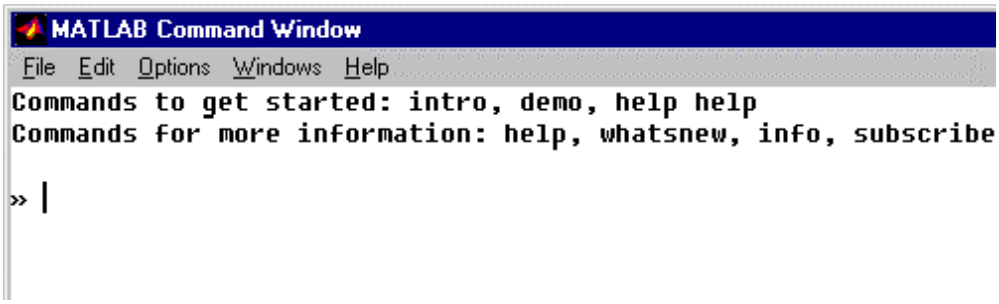


Рис. П1.2. Вызов пакета MATLAB

После появления на экране символа приглашения " >> " можно вводить команды пакета. Ввод команды завершается нажатием клавиши **ENTER**. Команды **exit**, **quit** приводят к завершению сеанса. Прерывание любых вычислений осуществляется нажатием клавиш **CTRL+C**.

Демонстрационная программа DEMO и справочник HELP

Начинать знакомство с пакетом **MATLAB** мы советуем с чтения электронного учебника (и демонстрационной программы одновременно) **Demo** и использования, по мере надобности, справочника **Help.** Введите команду **demo**: " >> **demo** " и перед вами откроется окно «**Welcome to the MATLAB Expo**». Следуя дальнейшим указаниям программы, вы сможете на примерах познакомиться с языком и библиотеками пакета.

Внимание! MATLAB *различает (режим, заданный по умолчанию) заглавные и строчные буквы.*

Краткая справочная информация о синтаксисе языка и особенностях применения встроенных функций пакета может быть получена с помощью команды **help.** Справочник пакета иерархический. По команде **help** на экран выводится только оглавление справочника. Если уточнить команду и указать дополнительно имя раздела из оглавления — **help <имя раздела>**, то можно получить краткую информацию о командах раздела. Указав имя конкретной команды, например: >>**help plot**, вы получаете полную информацию о ней.

Из всего многообразия языковых конструкций, существующих в пакете, мы остановимся только на тех, что нужны для выполнения приведенных в приложении учебных заданий. Эти конструкции можно разделить на две группы: конструкции, обеспечивающие матричные вычисления, и команды, с помощью которых можно построить графики в декартовой или полярной системе координат для функций одной или двух переменных (y = f(x), z = f(x,y)).

Матричные выражения

Решим с помощью пакета следующую задачу. Дана система линейных алгебраических уравнений

$$\mathbf{A*x = b}$$

с заданной квадратной вещественной матрицей **A** и вектором **b**. Предположим, что мы знаем точное решение системы $\mathbf{x_T}$. Требуется, используя возможности пакета, найти решение системы **x** — оно будет приближенным вследствие ошибок округления и вычислить погрешность $\|x_T - x\| / \|x_T\|$.

В языке пакета **MATLAB** существует один основной тип данных — прямоугольная (n, m)-матрица с комплексными коэффициентами. Любой элемент $a_{i,j}$ матрицы в памяти машины всегда представляется в виде двух ячеек (Re $a_{i,j}$, Im $a_{i,j}$), содержащих действительную и мнимую части, каждая из которых хранит число с плавающей точкой двойной точности (в терминах FORTRAN это числа типа Real*8). В то же время, числа на экране могут выглядеть как целые, как числа

с плавающей и фиксированной точкой. Формой представления чисел на экране можно управлять с помощью команды **format < имя формата >** (см. табл. П1.1).

Таблица П1.1

Команда format

format	**short**	фиксированная точка, 5 знаков
	long	фиксированная точка, 15 знаков
	short E	плавающая точка, 5 знаков
	long E	плавающая точка, 15 знаков
	HEX	шестнадцатеричные числа
	+	число, отличное от нуля, изображается знаком ("+"), нулевые числа — пробелом

Команда **format** также включена и в главное меню пакета. Для того чтобы управлять формой представления числа, достаточно найти в главном меню команду "Options", выбрать подкоманду "Numerical Format" и установить нужный режим представления чисел.

Элементы матриц можно записывать в любой форме — в виде целого числа, числа с фиксированной или плавающей точкой. Комплексные числа вводятся в виде арифметического выражения, например: **1+i** или **1+2*i**, где **i** — символ, изображающий мнимую единицу, а само выражение не должно содержать пробелов.

Матричные объекты могут иметь имена, которые с точки зрения синтаксиса ничем не отличаются от имен фортрановских переменных. Математической нотации

$$\mathbf{A} = \begin{vmatrix} \mathbf{1} & \mathbf{2} & \mathbf{3} \\ \mathbf{4} & \mathbf{5} & \mathbf{6} \\ \mathbf{7} & \mathbf{8} & \mathbf{9} \end{vmatrix},$$

определяющей матрицу A через значение ее коэффициентов, в **MATLAB**е соответствует следующая команда:

A = [1 2 3
4 5 6
7 8 9]
[ENTER].

Разделителем элементов в строке служит пробел, разделителем строк — невидимый символ конца строки. Строки также можно разделять с помощью символа "**;**" (точка с запятой):

A = [1 2 3 ; 4 5 6 ; 7 8 9] [ENTER].

Символом переноса строки служит " **...**" (троеточие).

A = [1 ...
2 ...
3 ...
4 5 6 ; 7 8 9] [ENTER].

Знак "=" (равно) трактуется как знак присваивания.

Примеры.

Вектор-строка:

X = [1 2 3] [ENTER].

Вектор-столбец:

Y = [1
2
3] [ENTER]

или

Y = [1; 2; 3] [ENTER].

Комплексная матрица:

A = [1+2*i 2+3*i ; 4+5*i 6+7*i] [ENTER]
A = [1 2 ; 4 6] + [2 3 ; 5 7] * i [ENTER].

или

ReA = [1 2 ; 4 6] [ENTER]
JmA = [2 3 ; 5 7] [ENTER]
A = ReA + JmA *i [ENTER].

Внимание! Если матричное выражение представлено в виде

A = [1 2 3] [ENTER],

то MATLAB *воспроизводит на экране введенный объект с учетом оператора* format. *Для подавления вывода используется символ* ";":

A = [1 2 3] ; [ENTER].

Далее операторы будут приводиться в виде

```
A = [ 1  2  3 ];
```

без очевидного указания [ENTER].

Во всех приведенных примерах элементами матриц служили числа. Вместо чисел MATLAB позволяет подставлять также любые матричные выражения, но результатом подстановки должна быть прямоугольная матрица.

Пример:

```
A = [ 1  2 ; 3  4 ];
C = [ 5  6 ] ;
D = [ 7 ; 8 ] ;
B = [ A  D ; C  9 ]
B = [ 1  2  7
      3  4  8
      5  6  9 ].
```

Символ " ' " (кавычка) употребляют как знак транспонирования (вещественной матрицы) или комплексного сопряжения.

Пример:

```
A = [ 1  2 ; 3  4 ];
C = [ 5  6 ];
C1 = C '  ;
B = [ A  C1  ; C  9 ]
B = [ 1  2  5
      3  4  6
      5  6  9 ].
```

В заключение заметим, что любые введенные числа считаются матрицами размером (1x1). Но для сохранения привычной формы за-

писи матрицы (1x1), т. е. числа, можно записывать без скобок: X = 2.0, а не X = [2.0].

Символом "[ ]" обозначаются неопределенные матрицы, то есть матрицы не имеющие ни размеров, ни значения. После присваивания A=[ ] теряется информация и о размерности, и о значениях матрицы A.

Чтобы лучше понять, что такое неопределенная матрица "[ ]", проделайте следующее. Создайте любую матрицу A и выберите в основном окне пункт меню File/Show Workspace. Перед вами откроется окно, показывающее содержимое рабочего пространства, где будет представлена матрица A. После присваивания A=[ ] матрица A из рабочего пространства исчезает.

Операции над матрицами

Над матрицами разрешены следующие операции.

1. Инфиксные арифметические **(**табл. П1.2).

Таблица П1.2

Арифметические операции

+	сложение
-	вычитание
*	умножение
/ (\)	левое (правое) деление
^	возведение в степень

2. Инфиксные логические **(**табл. П1.3).

Таблица П1.3

Инфиксные логические операции

&	AND	C = A & B;	1, если ai,j и bi,j отличны от нуля, Ci,j = 0, в противном случае
|	OR	C = A | B;	1, если ai,j или bi,j отличны от нуля, Ci,j = 0, если ai,j и bi,j оба равны нулю
~	NOT	C = ~A;	1, если ai,j=0, Ci,j= 0, если ai,j не равно нулю

3. Операции отношения (табл. П1.4).

Таблица П1.4

Операции отношения

<	меньше
< =	меньше равно
>	больше
> =	больше равно
= =	эквивалентно
~ =	не эквивалентно

Пример:

```
A=[ 1 2 ; 3 4 ];
B=[ 2 3 ; 4 5 ];
T=A == B
T= 0 0
   0 0
T=A < B
T= 1 1
   1 1
```

4. Префиксные, поэлементно выполняемые операции, соответствующие математическим функциям (табл. П1.5).

Таблица П1.5

Префиксные поэлементно выполняемые операции

abs	Абсолютное значения
sqrt	Корень квадратный
real	Вещественная часть числа
imag	Мнимая часть числа
conj	Комплексно-сопряженное число
round	Округление до ближайшего целого
fix	Округление по направлению к нулю
floor	Округление по направлению к минус бесконечности
cell	Округление по направлению к плюс бесконечности

Окончание табл. П1.5

sign	Знак числа
rem	Остаток от деления (для целых чисел)
sin	Синус
cos	Косинус
tan	Тангенс
asin	Арксинус
acos	Арккосинус
atan	Арктангенс
sinh	Синус гиперболический
cosh	Косинус гиперболический
tanh	Тангенс гиперболический
exp	Экспонента
log	Логарифм по основанию e
log10	Логарифм по основанию 10

5. Префиксные операции для построения матричных функций (табл. П1.6).

Таблица П1.6

Префиксные операции для построения матричных функций

expm(A)	Матричная экспонента
logm(A)	Матричный логарифм
sqrtm(A)	Матричный квадратный корень

6. Префиксные операции для построения матриц специального вида.

Таблица П1.7

Префиксные операции для построения матриц специального вида

diag(A)	Диагональная матрица, образованная из диагональных элементов матрицы A

Окончание табл. П1.7

eye(A) или **eye(n)**	Единичная матрица такой же размерности, что и матрица А единичная матрица размерности n*n
ones(A) или **ones(n)**	Все элементы матрицы такой же размерности, что и матрица А — единицы
zeros(A) или **zeros(n)**	Все элементы матрицы такой же размерности, что и матрица А — нули
rand(n)	Все элементы матрицы размерности n*n — случайные, равномерно распределенные числа из интервала [0,1]
magic(n)	Матрица размерности n*n, у которой сумма элементов строки равна сумме элементов соответствующего столбца
tril(A)	Выделить нижнюю треугольную матрицу из матрицы А
trul(A)	Выделить верхнюю треугольную матрицу из матрицы А

Для удобства пользователей введены префиксные операторы, помогающие инициализировать или строить матрицы специального вида (табл. П1.7).

7. Числа обусловленности, нормы и другие функции **(**табл. П1.8).

Таблица П1.8

Числа обусловленности, нормы и другие функции

cond(A)	число обусловленности матрицы по отношению к задаче обращения
rank (A)	ранг матрицы (находится численно)

Окончание табл. П1.8

rcond (A)	1 / cond — обратное к числу обусловленности
[n,m]= size(A)	Возвращает **n** — число строк матрицы, **m** — число столбцов матрицы
det(A)	определитель матрицы
матричные нормы	
norm(X,)	2-норма
norm(X, 1)	1-норма
norm(X, 2)	2-норма
norm(X, inf)	бесконечная норма
norm(X,'fro')	F-норма
векторные нормы	
norm(V, p)	p-норма
norm(V)	2-норма
norm(V, inf)	Максимальная по модулю компонента вектора
norm(V,-inf)	Минимальная по модулю компонента вектора

8. Матричные разложения (табл. П1.9)

Таблица П1.9

Матричные разложения

R=chol(A)	Построение разложения Холесского для матрицы A. Матрица R — верхняя треугольная, такая, что R'*R=A
[V,D]= eig(A)	Нахождение собственных векторов и собственных чисел матрицы A A*V=V*D
[V,D]= eig(A,'nobalance')	Нахождение собственных векторов и собственных чисел матрицы A, но без предварительного масштабирования матрицы
[P,H]= hess(A)	Приведение матрицы A к хессенберговой форме. A=P*H*P', P*P'=E

Окончание табл. П1.9

AINV= inv(A)	Построение обратной к A матрицы
[L,U]=lu(A)	L-U разложение матрицы A
A1=pinv(A)	Построение псевдо-обратной к A матрицы
[Q,R]=qr(A)	QR — разложение матрицы A. A=Q*R, Q — унитарная, R — верхняя треугольная
[U,T]=schur(A)	Разложение Шура. A=U*T*U', U — унитарная, T — верхняя треугольная
U=rref(A)	Ступенчатая форма матрицы A
Q=orth(A)	Ортогональный базис пространства столбцов матрицы A
Q=null(A)	Ортогональный базис ядра матрицы A
[U,S,V]=svd(A)	Сингулярное разложение матрицы A A=U*S*V', U, V — унитарные, S — диагональная
[L,U,P]=lu(A)	L-U разложение матрицы A с перестановками. Матрицы L, U, P такие, что P*A=L*U, P*P'=E, L — нижняя треугольная, U — верхняя треугольная матрицы

9. **И**нфиксные арифметические поэлементные операции (табл. П1.10).

Таблица П1.10

Инфиксные арифметические поэлементные операции

.*	поэлементное умножение
.^	поэлементное возведение в степень
./ или **.**	поэлементное деление
.'	транспонирование без комплексного сопряжения

Например:

```
a=[1-i 2+i];
b=a.'
b=[1-i
    2+i]
b1=a'
b1=[1+i
     2-i]
a=[1 2;3 4].^2
a=[ 1 4
    9 16]
```

10. Обработка экспериментальных данных и сигналов.

Для обработки экспериментальных данных (например, поиск max компоненты вектора, математического ожидания и т. д.) и обработки сигналов (преобразования Фурье, фильтрация данных и т. д.) воспользуйтесь возможностями библиотек (Toolboxes) с соответствующими именами.

Теперь мы способны решить систему A*x = b, используя возможности пакета. Приведенный ниже текст программы снабжен комментариями (символ " % " в начале строки):

```
% Сформировать матрицу A
A     = [2 3 ; 1 4];
%   вектор b
b     = [2 ; 6];
% и вектор x — известное заранее решение
x      = [-2 ; 2];
% Вычисление обратной матрицы
A_inv = inv(A);
% Нахождение y — приближенного решения
y      = A_inv*b;
% Вычисление нормы ошибки и вывод полученного значения
norm(y-x,1)
```

Результатом работы программы будет величина нормы: ans = 4.4409e-016.

Работа с элементами матриц

Как и в ФОРТРАНЕ, элемент произвольной матрицы A записывается в виде A(i,j). Очень часто при работе с квадратными (прямоугольными) матрицами необходимо выделить строку, столбец, либо подматрицу заданных размеров. Для этого требуется уметь описывать регулярное множество индексов. Аналогичная ситуация возникает при формировании значений компонентов векторов, носящих регулярный характер. Для описания регулярных множеств индексов используют символ " **:** ". Так, запись **X = [1 2 3 4 5]** эквивалентна записи **X = 1 : 5.** Регулярными могут оказаться и множества вещественных чисел. Например, шаг изменения переменных может быть вещественным положительным или отрицательным числом:

Y = 0 : 0.1 : 1;

Z = 1 : - 0.1 : 0.

Эти же обозначения используются и для формирования заданного множества индексов и, тем самым, для построения подматриц: **A(1 : 5, 1)**. Если исходная матрица имела размерность 5x5, то A(1 : 5, 1) — это подматрица вида

[A (1, 1)
A (2, 1)
A (3, 1)
A (4, 1)
A (5, 1)]

С помощью символа " **:** " (двоеточие) можно выделить строку **A (1, :)**, столбец **A (:, 1)**, представить двумерную матрицу A размерности nxn в виде вектора столбца **W = A (:)**, в котором элементы матрицы расположены столбец за столбцом. Допустимо выражение: **A (:) = 1 : 6.** Если A — это 3x2 матрица, то результатом будет матрица вида:

A = [1 4
2 5
3 6].

Графические возможности пакета MATLAB

Проиллюстрируем графические возможности пакета на примере следующей задачи. Пусть существует система ОДУ:

$$\frac{dx}{dt} = A*x + b,$$

где **t** принадлежит интервалу **[0,T].** Пусть заданы матрица **A**, вектор **b**, вектор начальных условий **x0 = x(0)** и вектор **x_т (t)** — известное аналитическое решение исходной системы. Требуется численно найти вектор **x(t)** — приближенное решение системы, оценить норму

$$|| \mathbf{x(t) - x\_т(t)} ||$$

и построить графики зависимости этих величин от времени.

Для решения этой задачи нам понадобятся основные команды для построения графиков (табл. П1.11), директивы (табл. П1.12) и средства «редактирования» графиков (табл. П1.13)

Таблица П1.11

Команды для построения графиков

plot	Построение графика одномерных функций в декартовых координатах
loglog	Выбор логарифмического масштаба по обеим переменным
semilogx	Выбор логарифмического масштаба по оси X
semilogy	выбор логарифмического масштаба по оси Y
polar	Построения графика в полярных координатах
mesh	Построения графика двумерных функций
contour	Построение линий уровня для двумерных функций
axis	«Ручное» масштабирование осей графика
clabel	Разметка линий уровня
grid	Нанесение градуировочной сетки на график

Таблица П1.12

Директивы

shg	Показать графический экран
clg	Очистить графический экран
clg	Сохранить на экране текущее изображение
prtsc	печать содержимого графического экрана

Таблица П1.13

Средства «редактирования» графиков

title	Создать заголовок графика
xlabel	Подписать X-ось
ylabel	Подписать Y-ось
text	Поместить текст в указанную область графического экрана

Разберем возможности команды **plot**. В простейшем случае, если y — вектор-столбец (строка), команда **plot (y)** означает по умолчанию, что абсциссы x функции y(x) равны последовательности целых чисел **x = [1 2 ... n];** где n — число координат вектора y. При явном задании вектора x, команда имеет вид: **plot (x, y) ;**. График будет построен с автоматическим выбором масштаба по осям. Если написать фрагмент программы:

```
x = [ 1 2 3 4 5 ] ;
y = [ 1 2 3 4 5 ] ;
plot ( x,y );
title ('график')
xlabel ( 'x - ось ');
ylabel ( 'y - ось') ;
grid ;
```

получим график, у которого есть названия осей и заголовок (рис. П1.3).

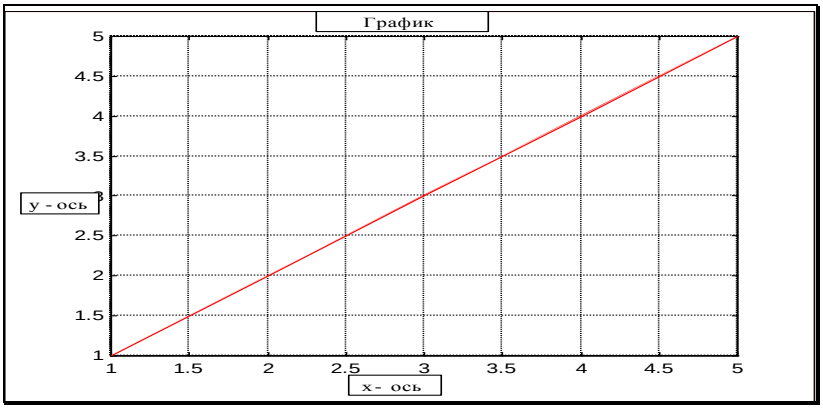


Рис. П1.3. Построение графика

Таблица П1.14

Средства разметки графиков

СИМВОЛЫ РАЗМЕТКИ	ЦВЕТ
- **--** **:** **-.** **.** **+** ***** **o** **x**	r — красный g — зеленый b — голубой w — белый i — график не отображается

Можно управлять цветом кривых и помечать их какими-либо специальными символами: plot (x1, y1, " + ", x2, y2, " - "); или plot (x1, y1, 'r', x2, y2, 'g'). В первом случае рисуется два графика. Их цвет выбирается автоматически, первый график помечается символом "+", а второй — символом "–". Во втором случае также строятся два графика. Но теперь первый график будет нарисован красной линией, а второй — зеленой. Для разметки графиков можно использовать символы, указанные в табл. П1.14.

Как работает простейшая 3D-графика, поясним на примере. Выполним команды:

Z = eye(10) ;

mesh (Z) ;

и получим график, изображенный на рис. П1.4.

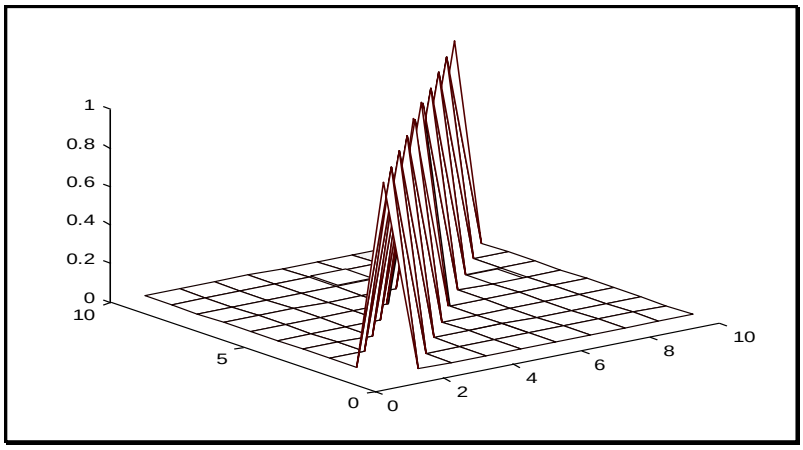


Рис. П1.4. 3D-график

Как и в случае графиков для функций одной переменной можно отказаться от автомасштабирования — в этом случае необходимо явно указать значения x_min, x_max, y_min, y_max: **axis (x_min, x_max, y_min,y_max);**. На одном экране можно отобразить несколько графических окон одновременно. В этом случае надо явно указать, где будет находиться график на экране, воспользовавшись командой **subplot**. Например**,** в команде **subplot (nij)** количество экранов n и их координаты ij кодируются трехзначным числом nij, в котором первая цифра числа — количество экранов, вторая и третья — координаты экрана. В случае если выдаются команды: **subplot** (211) и **subplot** (212), то число экранов — 2, их номера — 11 и 12 и весь экран будет выглядеть так, как показано на рис. П1.5.

211
212

Рис. П1.5. Вид экрана

Следующая последовательность команд отображает графики в различных графических окнах:

```
subplot(211);
plot ( x1, y1, " + ", x2, y2, " - " );
subplot(212);
plot ( x1, y1, 'r', x2, y2, 'g' ) ;
```

Примечание. *Для печати графиков необходимо предварительно запустить драйверы graphics.com (DOS-файл) или egaepson.com (C:\MATLAB\bin)*

Более подробно о графических возможностях **MatLab** *вы можете узнать, воспользовавшись информацией, предоставляемых* **Demo/Help**.

В качестве примера приведем решение задачи о построении и визуализации решения системы дифференциальных уравнений:

```
% Построение вектора начальных условий
Y0 = [1;1];
%  Выбор интервал [0 T] интегрирования
TSPAN = [0 20];
% Решение системы ОДУ методом Рунге–Кутта
[t,Y] = ode45 ('fun1',TSPAN,Y0);
% Вычисление нормы вектора x(t) — x_т(t)
for i = 1 : length(t),
    z(i) = norm(Y(i) - exp(t(i)), 1);
end;
%  Построение графиков
plot (t, Y, 'r');
pause;
plot (t, exp(t), 'b');
pause;
plot (t, z, 'g');
```

Подпрограмма fun1.m, вызываемая стандартной подпрограммой ode45, вычисляет правую часть и может быть представлена следующим образом:

```
function [S] = fun1(t,X)
% В данном случае матрица A формируется в подпрограмме
A = eye (2);
% Ка и вектор b
b = [0 ; 0];
% В общем случае их можно передать из вызывающей программы
% с помощью оператора global
% описание правой части
S = [0 ; 0];
S(1) = A(1,1)*X(1) + A(1,2)*X(2) + b(1);
S(2) = A(2,1)*X(1) + A(2,2)*X(2) + b(2);
return;
```

В результате на экран будут выведены графики функций x(t), **x_т(t)** и нормы вектора ошибки **(x(t) - x_т(t))**, как функции от времени**.**

Элементы программирование в пакете MATLAB

Последовательность команд пакета **MATLAB** можно рассматривать как обычную программу, записанную на специальном алгоритмическом языке. Такую программу можно подготовить заранее, как текстовый файл, а затем выполнить ее. Единственное отличие от подготовки программ на других языках состоит в том, что программа должна быть размещена в так называемом m-файле.

Рассмотрим подробнее работу с m-файлами.

Средствами любого редактора создайте файл с расширением "name**.m**". Так, например, выбрав в MATLAB Command Window команду **File/ New/ M-file** мы создаем m-файл в редакторе **Notepad.** В этом окне можно создавать новый и редактировать существующий m-файл. Для редактирования файла необходимо выбрать в MATLAB Command Window команду **Open M-file...** и в появившемся окне уста-

новить путь к соответствующему файлу, то есть выделить его мышью и нажать **ОК**. Для сохранения m-файла достаточно в активном окне редактора выбрать в меню команду **File/Save** и указать место сохранения файла на диске. Если созданный m-файл находится в той же директории, что и сам **Matlab**, то для запуска m-файла в этом случае достаточно написать имя этого файла в окне Command Window и нажать **enter**. Например, если существует m-файл work1.m, то его запуск осуществляется командой **>>work1.** Если же m-файл находится в какой-либо другой директории, то для его запуска необходимо выбрать в MATLAB Command Window команду Run M-file.... В появившемся окне Run M-file нажмите кнопку **Browse**. В окне **Browse** установите путь к необходимому m-файлу, выделите его и нажмите ОК. В результате этих действий вы возвратитесь в окно Run M-file, но с уже прописанным путем к необходимому файлу. Теперь достаточно для запуска m-файла нажать кнопку ОК.

Примечание. *Для получения более подробной информации о той или иной возможности пакета воспользуйтесь командами* **Demo** *и* **Help**.

Управляющие конструкции

Операторы циклов. В MATLABe существует два вида циклов: **for** и **while**. Цикл **for** имеет вид:

```
for v = выражение
        оператор;
end
```

Если вы хотите записать этот оператор одной строкой, необходимо использовать запятую в качестве разделителя:

```
for v = выражение,    оператор;    end
```

Для того чтобы присвоить компонентам вектора конкретные значения, можно написать:

```
for   i = 1 : 10,
x ( i ) = i;
end
```

Обратите внимание, что мы поставили лишнюю запятую, но это допустимо. Примером использования двойного цикла может быть следующий фрагмент:

```
for    i = 1 : n ,
for    j = 1 : n ,
    a ( i,j ) = 1 / ( i + j );
end
end
```

Не забывайте, что точка с запятой подавляет вывод на экран значений вычисленных выражений, что особенно важно при работе с циклами.

В цикле можно использовать матричное выражение:

```
for  V = A,
оператор;
end
```

которое трактуется следующим образом. Переменной V присваиваются значения матрицы A, столбец за столбцом, а далее выполняется цикл. Например, суммировать элементы матрицы можно так:

```
E = eye(10);
[ m, n ] = size ( E );
s=0;
for j = 1 : n;
for V = E ( :, j ),
s=s+sum(V);
end ;
end;
```

Цикл **while** выглядит традиционно:

```
 while       условие,
 оператор;
end
```

или

```
while  условие, оператор;  end
```

Если в качестве условия стоит матричное выражение

while A>=0 , оператор; end

то цикл будет выполнятся до тех пор, пока все элементы матрицы A не станут нулями, т. е. значение условного выражения окажется равным **false**. Например:

```
E = eye(10);
j=0;
s=0;
while A>=0,
j=j+1;
for V = E ( :, j ),
s=s+sum(V);
end
E(j,j)=0;
end
```

Конструкция выбора

Конструкция выбора записывается в виде:

if условие,
оператор;
elseif условие,
оператор;
else
оператор;
end

Примером использования конструкции выбора может служить следующий фрагмент программы:

```
n=9;
if n==7,
v=5;
elseif  n==8,
v=6;
else
v=7;
end;
```

Подпрограммы

Предположим, что мы хотим создать подпрограмму вычисления евклидовой нормы вектора NORM (X). Средствами любого редактора создаем файл с расширением **.m** и именем, совпадающим с именем функции, например **norm. m.** В этот файл помещаем текст подпрограммы:

```
function  r = norm(x)
% Пример подпрограммы
r =sqrt(x * x' ) ;
```

Конец подпрограммы совпадает с невидимым пользователю символом конца файла. Теперь для вычисления нормы достаточно написать:

```
z = [ 1 2 3 ] ;
a = norm(z);
```

Если выходных параметров несколько, то синтаксис первого предложения меняется и следует писать так: **function [X1, X2, X3, ... XN] = имя (Y1, Y2, ... YM),** где **Yi** — входные, а **Xi** — выходные параметры подпрограммы. Аналогом фортрановского блока COMMON служит оператор **global** *список переменных*, например, **global a b c.** Оператор помещается в тело процедуры, после чего перечисленные объекты списка становятся доступными в вызывающей программе.

Замечание. *Переменные списка должны быть определены к моменту объявления!*

Все переменные, используемые в теле m-файла, с точки зрения вызывающей программы являются локальными переменными. Входные переменные передаются значением, а не адресом, и, следовательно, в вызывающей программе не изменяются, даже если они были изменены в теле процедуры.

Дополнительные команды

Команда **clear** приводит к очистке рабочего пространства, удаляя все существующие в программе/подпрограмме переменные. Команда **clear** *имя вектора* удаляет указанный вектор.

Примечание. *Для иллюстрации результата операции* **clear** *удобно воспользоваться пунктом меню* File/Show Workspace. *Команда* **clear***, в отличие от* **[ ]***, не удаляет матрицу из рабочего пространства, а лишь освобождает ее от содержимого.*

Команда **clock** выводит полную информацию о текущем моменте времени:

clock = [год месяц день час мин. сек.].

Для подсчета времени выполнения операций рекомендуется использовать команды **tik, tok и cputime.**

Команда **flops** возвращает общее число операций с плавающей точкой, потраченных на выполнение той или иной команды. Сосчитаем число операций, необходимых для вычисления обратной матрицы:

% Создаем тестовую матрицу A

A = eye(2);

% Включаем счетчик числа операций

flops(0);

% Обращаем матрицу A

inv(A);

% Подсчитываем число операций и выводим полученный результат

flops.

Результатом работы программы будет величина:

ans = 33.

Приложение 2

MAPLE ДЛЯ НАЧИНАЮЩИХ

Основные сведения о пакете

Под символьными вычислениями обычно понимают нахождение результата в замкнутой форме. В настоящее время существует много программных комплексов, позволяющих производить символьные вычисления, а именно, Derive, Reduce, Maple, Mathematica и другие. Среди них Maple занимает особое место [7]. Помимо того, что Maple используется для вычислений непосредственно, он встраивается и в другие пакеты как составная и обычно невидимая пользователю часть. Так это сделано, например, в пакете MathCad. Форму представления данных в MathCad можно назвать графической. Математические выражения предстают в нем перед пользователем, как говорят программисты, в «дружественном» виде, такими, какими мы их видим в книгах, а уже потом «переводятся» на язык пакета Maple. Причина в том, что язык пакета Maple достаточно труден для восприятия. Однако изучение его языка позволяет лучше осознать все достоинства и недостатки символьных вычислений и перекинуть мостик практически к любой программной среде, где реализованы символьные вычисления.

Пакет Maple состоит из трех частей: ядра — процедур, написанных на языке C, библиотеки, написанной на Maple-языке, и интерфейса. Большинство базисных операций выполняется ядром. Специализированные команды содержатся в библиотеке. Пользователь может расширять возможности языка Maple, создавая свои процедуры и функции.

Основное назначение всех символьных пакетов — преобразовывать выражения, анализировать свойства функций и решать различные уравнения в замкнутой форме. Помимо символьных вычислений, все известные символьные пакеты могут проводить и численные расчеты, но последние лучше выполнять на ФОРТРАНе. Maple позволя-

ет: проводить исследование функций, применяя аппарат математического анализа, а именно, дифференцировать, интегрировать, находить пределы, вычислять суммы, произведения, строить и суммировать различные ряды, интерполировать и аппроксимировать функции, работать с полиномами, в частности, ортогональными, строить интегральные преобразования, находить решение систем линейных алгебраических уравнений, нелинейных алгебраических уравнений, в частности, искать корни полиномов, решать дифференциальные уравнения и системы дифференциальных уравнений аналитически и численно.

Первые сведения о командах

Работа с пакетом происходит в режиме интерпретации. В строке ввода пользователь набирает команду, нажимает клавишу **Enter** и получает результат, содержащийся в строке вывода. Это может быть формула, график, сообщение об ошибке. И вводимая информация, и результаты вычислений содержатся в одном документе — в файле, который можно сохранить для дальнейшего использования. Полученный документ является одновременно и программой, и отчетом о проделанной работе.

В качестве примера вычислим сумму квадратов целых чисел от 1 до 2000 и присвоим результат переменной A. Вызовем пакет. В открывшемся окне, после символа приглашения " >> ", наберем команду **A:=sum(k^2, k=1...2000);** и нажмем клавишу **Enter**. На экране появится результат вычислений: **A:= 2668667000**. Maple проверил правильность введенной команды и вычислил результат. В этом фрагменте программы были использованы: оператор присваивания " **:=** ", с помощью которого выражению **sum(k^2, k=1...2000)** было присвоено имя A (именно так трактуется оператор присваивания в Maple), встроенный оператор **sum**, вычисляющий сумму, и разделитель команд " **;** ". В отличие от Matlab, где данный разделитель подавляет вывод, в Maple он требует отображения результата на экран. Подавить вывод можно с помощью разделителя " **:** ". В результате вычис-

лений имя A оказалось присвоенным уже другому выражению, а именно, числу **2668667000**. Заметим, что Maple различает прописные и строчные буквы, так что **sum(k^2, k=1...2000)** и **Sum(k^2, k=1...2000)** — это не одно и тоже.

Другим примером является построение графика y = sin(x), где x меняется от -π до π. Введите команду: **plot(sin(x),x=-Pi..Pi);** нажмите **Enter.** Сравните синтаксис данной команды с аналогичной командой пакета **Matlab**. Попробуйте построить график функции sin(x)/x или 1/((x-1)*(x+1)) на интервале от -π до π. Обратите внимание на то, как пакет справляется с особыми точками.

Краткие сведения о командах главного меню и справочной информации

Главное меню. При вызове программы Maple появляется типовое окно Windows, которое содержит Основное меню, Панель инструментов (набор иконок) и Строку состояния (внизу экрана). В основном окне открывается новый документ с именем Untitled. Именно в нем набираются, вычисляются и отображаются результаты вычислений.

Основное меню состоит из восьми пунктов (табл. П2.1).

Таблица П2.1

Структура основного меню

File	Поиск, чтение и запоминание файлов
Edit	Редактирование файла
View	Изменение режимов отображения документа
Insert	Вставка в документ фрагментов других документов
Format	Форматирование документа
Options	Настройки пакета
Window	Управления окнами, видимыми на экране
Help	Справочная информация

В меню работы с файлами **File** нам понадобятся следующие подпункты (см. табл. П2.2).

Таблица П2.2

Подпункты меню File

New	Ctrl-N	Начало работы с новым файлом
Open	Ctrl-O	Считывание файла с диска
Save	Ctrl-S	Сохранение текущего файла на диске
Save as...		Сохранение файла под новым именем
Exit	Alt-X	Выход из программы

Во второй колонке табл. П2.2 указаны комбинации так называемых «горячих» клавиш, нажатие на которые приводит к исполнению закрепленных за ними команд.

В меню редактирования **Edit** входят следующие команды (см. табл. П2.3).

Таблица П2.3

Подпункты меню Edit

Cut	**Ctrl-X**	Удаление выделенного текста с сохранением в буфере
Copy	**Ctrl-C**	Копирование выделенного текста в буфер
Paste	**Ctrl-V**	Копирование текста из буфера, начиная с позиции, указанной курсором
Delete	**Delete**	Удаление выделенного текста
Execute **Selection** **Worksheet**		Вычисление выделенного фрагмента документа всего документа

Начиная работать с пакетом, советуем Вам выполнить следующие команды меню **View** (поставить «галочку» рядом с соответствующими именами) (см. табл. П2.4).

Справочная информация

Одним из самых важных и необходимых для успешной работы является подменю **Help** (Помощь или Справка). Используя Help,

можно узнать, какие команды есть в системе, уточнить их синтаксис, посмотреть примеры использования незнакомой команды и многое другое. Отображаемое окно Help представляет собой гипердокумент. В нем есть ссылки на другие части документа (они выделены подчеркиванием и цветом), активизировав которые с помощью мыши, можно получить в другом окне справочную информацию по выбранной теме. Основные команды меню Help приведены в табл. П2.5.

Таблица П2.4

Подпункты меню View

Tool Bar	Вывести на экран строку иконок, помогающих быстрее работать в среде **Maple**. Иконкам соответствуют операции: открытие нового документа, запись документа в файл, печать документа на принтере; удаление, копирование, вставка выделенного фрагмента документа; остановка вычислений и др.
Context Bar	Вывести на экран строку иконок, появляющихся в зависимости оттого, что вы делаете в данный момент. Если вы набираете текст, то она содержит иконки, соответствующие командам проверки правильности синтаксиса команды, исполнения команды и др.; если вы активизировали (выделили) график, то появляются иконки, меняющие вид графика: перенос графика в начало координат, точечное отображение графика, линейное отображение графика (даже если отображается набор дискретных величин) и др.
Status Bar	Вывести на экран строку состояния программы. Она отображает информацию о выбранной команде меню (в нижнем углу экрана); времени выполнения команд; объеме памяти, отведенной под переменные, объеме свободной памяти (последние три пункта видны в правом нижнем углу экрана)

Таблица П2.5

Основные команды меню Help

Contents	Доступ к оглавлению Справки. Из него можно дальше попасть в конкретный раздел или параграф
Help on Context (Ctrl-F1)	Контекстная справка. Одна из самых необходимых команд. Допустим, вы помните только имя команды. Наберите имя команды, зафиксируйте на нем курсор ввода текста и нажмите клавиши **Ctrl-F1.** В результате этих действий на экране появится нужная вам информация. Аналогичный результат дает выполнение команды **?имя_команды**, например, **?plot,** набранной в строке ввода
History	Обращение к уже вызывавшимся фрагментам Справки. Удобство состоит в том, что не нужно заново просматривать множество ссылок в гипертексте. Достаточно выбрать соответствующий элемент из списка **History** и получить требуемую справку. Советуем Вам дополнительно включить опцию (поставить галочку) **Balloon Help** — это позволит отображать краткую помощь о команде меню прямо около курсора мышки в более удобном виде

Основные конструкции языка

Единственным объектом, с которым умеет работать Maple, является математическое выражение (**expression**). Существуют свои правила написания алгебраических, арифметических, логических выражений и выражений других различаемых пакетом типов.

Узнать тип выражения можно, используя команду (с точки зрения пакета это выражение типа «функция») **whattype(expr)**, где в качестве параметра ей передается произвольное выражение, а она возвращает в качестве значения его тип. Тип выражения проверяется с помощью команды **type(x,t)**, где **x** — выражение, а **t** — название типа. Команда возвращает значение «истина» (**true**), если выражение **x**

имеет тип **t**, или «ложь» (**false**), если **x** не является типом **t.** Знать тип выражения чрезвычайно важно, так как от этого зависит результат вычислений.

В Maple существуют и более сложные, чем математические выражения, объекты. К ним относятся структурированные типы — таблицы, списки, множества, векторы и матрицы.

Разнообразные математические выражения следует отличать от других конструкций языка программирования, связанных с классическим процедурным программированием. К последним относятся: операторы присваивания, условные операторы, операторы цикла, прерывания циклических вычислений, ввода-вывода, операторы описания и вызова процедур и останова. Математические выражения рассматриваются как вызовы функций. Для того чтобы убедиться в этом, рассмотрим еще одну команду пакета **op(expr)** и связанную с ней команду **nops(expr)**. Операция **op(expr)** анализирует структуру выражения и может, в зависимости от формы обращения, либо указывать тип выражения **expr (op(0, expr))**, либо возвращать i-тый «аргумент» выражения **(op(i, expr))**. В качестве примера рассмотрим выражение a + b. Зададим вопрос о структуре выражения:

> **op(0, a+b);**

+

> op(1, a+b);

a

> op(2, a+b);

b

Ответы пакета легко понять, если переписать исходное выражение в форме a+b=+ab=+(a,b).

Операция **nops(expr)** возвращает число аргументов выражения:

> nops(d);

1

> op(0,d);

string

> op(1,d);

d

В данном примере мы попросили пакет ответить на вопросы, сколько аргументов в выражении d, и какой у него тип. Так как мы ввели только имя переменной, то пакет считает, что имеет дело со строкой.

Выражения

Самым простым типом математических выражений считается имя. Имена могут быть присвоены конкретным выражениям, и тогда они используются как синонимы этих выражений. Имена могут существовать и сами по себе, и тогда они трактуются как переменные. Имена должны обязательно начинаться с буквы. С именами связана конструкция **name:='name'**, поначалу кажущаяся абсурдной, однако она очень точно отражает суть дела. С помощью этой фразы, мы говорим, что имя **name** теперь обозначает только переменную с именем **name** и не имеет никакого конкретного значения. Именно поэтому имена имеют тип «строка» — **string**.

За частью имен закреплены конкретные значения, и тогда они называются константами (**constant**), точнее, защищенными именами (табл. П2.6).

Таблица П2.6

Значения констант

Pi	число π
E	число e (основание натурального логарифма)
I	мнимая единица
infinity	∞ (бесконечность)
gamma	константа Эйлера
true	истина (логическая константа)
false	ложь (логическая константа)

Значения этих величин не могут быть изменены. Наряду с математическими константами существуют зарезервированные имена языка программирования, трактуемые как управляющие переменные, например, **Digits** и **Order**. Первая (**Digits**) отвечает за количество зна-

ков участвующих в вычислениях чисел с плавающей точкой. Значение по умолчание равно **десяти**. Увеличивая данное значение, мы повышаем точность, но при этом возрастает время выполнения вычислений. Вторая управляющая переменная (**Order**) отвечает за максимальное число выводимых на экран членов при вычислении рядов. Значение по умолчанию равно **шести**.

Ниже приведены примеры, поясняющие использование управляющих переменных. В первом вычисляется корень из двух с различной точностью, а во втором экспонента разлагается в ряд в окрестности точки ноль, и выводится три и пять членов ряда соответственно:

> a:=sqrt(2):Digits:=2:evalf(a);Digits:=20:evalf(a);

> Order:=3:series(exp(x),x=0);Order:=5:series(exp(x),x=0);

$$1.4$$

$$1.4142135623730950488$$

$$1+x+\frac{1}{2}x^2+O(x^3)$$

$$1+x+\frac{1}{2}x^2+\frac{1}{6}x^3+\frac{1}{24}x^4+O(x^5)$$

Обратите внимание, что команда **a:=sqrt(2)** только присваивает имя выражению. Через **sqrt(2)** обозначено алгебраическое число. Вычисляется оно с помощью команды **evalf(a).**

Более сложным типом является алгебраическое выражение (тип "**algebraic**"). Это привычные всем математические формулы, содержащие числа (числа в пакете могут быть целыми, дробями, в форме с плавающей точкой), переменные, знаки действий и функции (инфиксные и постфиксные).

Вот несколько примеров алгебраических выражений и их конкретных типов:

> whattype(5);

integer

> whattype(1.4);

float

> whattype(1/2);

fraction

> type(1+I, complex);

true

> whattype(sin(x));

function

> type(1.4,integer);

false

> type(1/2*sin(1), algebraic);

true

> whattype(z+v);

+

> type(3>x,algebraic);

false

Среди алгебраических выражений выделяют арифметические выражения. При построении арифметических выражений, как обычно, можно использовать знаки: "**+**" — сложение, "**–**" — вычитание, "*****" — умножение, "/" — деление, "^" — возведение в степень, "**!**" — вычисление факториала. Вычисление выражения происходит с учетом приоритета операций: сначала производится возведение в степень, затем умножение и деление, а затем сложение и вычитание. Операции выполняются слева направо. Чтобы изменить порядок вычисления, требуется использовать круглые скобки.

В Maple, конечно, есть привычные всем элементарные функции. Перечислим наиболее распространенные (см. табл. 2.7).

Далее следуют тригонометрические функции (аргумент задается в радианах, **x** — алгебраическое выражение): **sin(x)**, **cos(x)**, **tan(x)**, **cot(x)**, **sec(x)**, **csc(x)**; обратные тригонометрические функции: **arcsin(x)**, **arccos(x)**, **arctan(x)**, **arccot(x)**, **arcsec(x), arccsc(x)**; гиперболические функции: **sinh(x)**, **cosh(x)**, **tanh(x)**, **coth(x)**, **sech(x)**, **csch(x)**; обратные гиперболические функции: **arcsinh(x)**, **arccosh(x)**, **arctanh(x)**, **arccoth(x)**, **arcsech(x), arccsch(x)**.

Таблица П2.7

Элементарные функции

Математическая запись	Запись на языке **Maple**
e^x	**exp(x)**
ln x	**ln(x)** или **log(x)**
$\log_{10} x$	**log10(x)**
$\log_a x$	**log[a](x)**
$\sqrt{x}$	**sqrt(x)**
$\lvert x \rvert$	**abs(x)**
sgn x	**signum(x)**
n!	**n!**

Еще один, свойственный всем языкам тип выражений — логический (**logical**). Логическое выражение может включать «отношения» (**relation**), булевские выражения (**boolean**) и булевские константы: «истина» (**true**) и «ложь» (**false**). Отношение — это выражение, построенное с помощью операторов: **=**, **<>**, **<,** или **<=**. Булевское выражение строится с помощью операторов: **and** (логическое И), **or** (логическое ИЛИ) или **not** (отрицание).

Существует еще несколько важных типов выражений. К ним относятся выражения типа «ряд» (**series**). Разложение в ряд осуществляется командой **series(expr, eqn, n)**; где **expr** — выражение, разложение которого ищется; **eqn** — точка, в окрестности которой строится ряд (по умолчанию вычисление происходит в окрестности нуля); **n** — число членов ряда (если значение n не указывается, то берется значение, присвоенное переменной **Order**). Поясним, как строятся ряды, на примерах:

> series(exp(x), x=0, 4);

> series(x/(1+x), x=1, 2);

$$1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + O(x^4)$$

$$\frac{1}{2} + \frac{1}{4}(x-1) + O((x-1)^2)$$

Maple признает рядами только разложения выражений относительно указанной переменной по целым степеням. Выражения с дробными степенями трактуются как суммы. Например:

> a := series(sin(x), x=0,7);type(a,series);whattype(a);

> b:=series(sin(x)^(1/2), x=0, 7);type(b,series);whattype(b);

$$a := x - \frac{1}{6}x^3 + \frac{1}{120}x^5 + O(x^7)$$

true

series

$$b := \sqrt{x} - \frac{1}{12}x^{\frac{5}{12}} + \frac{1}{1440}x^{\frac{9}{12}} + O(x^{\frac{13}{12}})$$

false

+

Заканчивая разговор о типах алгебраических выражений, укажем еще один практически важный тип. Это — последовательность выражений. Например:

> a:=1,4,34,-23,sin(r),k;

a:=1,4,34,-23,sin(r),k;

В такой последовательности можно выделить i-ый член a[i], например, *a*[1]=1.

К основным структурированным типам относятся: список (**list**) и множество (**set**). Множество — это неупорядоченная последовательность имен, заключенных в фигурные скобки **{ }**. Список — это упорядоченная последовательность имен, заключенных в квадратные скобки **[ ]**. И список, и множество могут быть пустыми. Можно выделять подмножество или подсписок. Пусть **S** — список или множество, тогда **S[i..j]** (или, что то же самое, **op(i..j,S)**) — это подмножество или подсписок, начиная с элемента **i** и кончая элементом **j**. Чтобы добавить элемент **x** к списку **L**, необходимо выполнить операцию **[op(L),x]**. Вставка элемента **x** в множество **S** будет выглядеть так: **S union {x}** (**union** — объединение). Замещение **i**-го элемента в списке **L** на **x**: **subsop(i=x,L)**. Удаление **i**-го элемента из списка **L**:

subsop(i=NULL,L). Удаление элемента **x** из множества **S**: **S minus {x}** (**minus** — разность). Для множества также определена операция пересечения **intersect**. Покажем это на примерах:

>L:=[1,2,6*w-cos(u),2,1];S:={1,sin(z),3,2,1};L1:=L[3..4];S1:=S[1..2]; L:=[op(L),10];

> S:=S union {5};L:=subsop(1=NULL,L);S:=S minus {2};

L := [1, 2, 6 w - cos(u), 2, 1]

S := {1, 2, 3, sin(z)}

L1 := [6 w - cos(u), 2]

S1 := {1, 2}

L := [1, 2, 6 w - cos(u), 2, 1, 10]

S := {1, 2, 3, 5, sin(z)}

L := [2, 6 w - cos(u), 2, 1, 10]

S := {1, 3, 5, sin(z)}

Еще несколько важных типов — массив (**array**), матрица и вектор — мы рассмотрим позже. Пока нам достаточно знать, что массив создается с помощью операции **array(bounds,list**), где **bounds** — выражение, описывающее границы индексов, а **list** — выражение типа список с указанными значениями массива. В массиве индексы задаются с помощью целочисленной переменной, принимающей значения из диапазона **n..m**. Выражение **n..m** — это специальное выражение типа «диапазон». Например, создадим одномерный массив

> d:=array(1..2);

d := array(1 .. 2, [])

Если снять требование, чтобы индексы были целочисленными, получим еще один тип данных — таблицы (table):

> x:=table(one, two);

x := table (one, two, [])

И, наконец, одномерные и двумерные массивы, индексы которых начинаются с единицы, называются векторами и матрицами.

Преобразование типов

В **Maple** существует команда **convert**, позволяющая преобразовывать выражение одного типа в другой, то есть менять его форму хранения, отображения и способ обработки. Общий вид функции преобразования типов:

convert(expr, form, arg3, ...),

где **expr** — произвольное выражение; **type** — тип преобразования; **arg3, ...** — дополнительные аргументы.

Перечислим основные возможности функции **convert** (см. табл. П2.8).

Таблица П2.8

Основные возможности функции convert

convert(object, array)	преобразование в массив; **object** — список, таблица, массив
convert(object, list)	преобразование в список; **object** — таблица, вектор, выражение
convert(A, matrix)	преобразование в матрицу; **A** — двумерный массив или список списков
convert(L, vector)	преобразование в вектор; **L** — список или одномерный массив
convert(object, table)	преобразование в таблицу; **object** — список или массив

Примеры:

>a:=[[1,2],[3,4]];b:=convert(a,matrix);

a := [[1, 2], [3, 4]]

Преобразование выражений

Над выражениями в Maple можно производить множество операций. К ним относятся: упрощения, выделение части выражения, различного рода подстановки. К наиболее часто употребляемым дей-

ствиям-преобразованиям над выражениями относятся приведенные в табл. П2.9.

Таблица П2.9

Преобразования над выражениями

simplify(expr)	Упрощение выражения **expr**
factor(expr)	Разложение выражения **expr** на множители
collect(a, x)	Приведение подобных членов в выражении **a** относительно переменной **x**
expand(expr)	Раскрытие скобок в выражении **expr**
subs(x=a,expr)	Подстановка подвыражения **x = a** в выражение **expr**
normal(e)	Приведение дроби ***e*** к нормальной форме
numer(e)	Выделение числителя из алгебраического выражения **e**
denom(e)	выделение знаменателя из алгебраического выражения **e**

Примеры:

> simplify(b*b+b*c);

$b^2 + b\,c$

> factor(a^2+2*a*b+b^2);

$(a + b)^2$

> simplify(sin(x)^2+cos(x)^2);factor(x^2+10*x-24);

> collect(x^4-3*x+12*x*y-y^2,x);expand((-x+2)/(x+2));

$$\frac{1}{(x + 12)\,(x - 2)}$$

$$x^4 + (-3+12\,y)x - y^2$$

$$-\frac{x}{x+2} + \frac{2}{x+2}$$

> subs(a=-1,a^2-5*a+2);l:=(x^2-y^2)/(x-y)^3;l:=normal(l);numer(l);denom(l);

8

$$l := \frac{x^2 - y^2}{(x - y)^3}$$

$$l := \frac{y + x}{(-y + x)^2}$$

$$y + x$$

$$(-x + y)^2$$

Выражения и вычисления

Все, что делает Maple, можно назвать вычислениями. Однако у термина «вычисления» есть и более узкие толкования.

Прежде всего, Maple умеет вычислять интегралы и производные в замкнутой форме. Например, если выражение рассматривается как функция от заданных аргументов, то можно вычислять производные от нее, интегрировать, раскладывать в ряд (см. табл. П2.10).

Таблица П2.10

Вычисления, выполняемые пакетом

diff(a, x1, x2, ..., xn)	Дифференцирование выражения. **a** — алгебраическое выражение; **x1, x2, ..., xn** — переменные, по которым происходит дифференцирование
int(f, x)	Вычисление неопределенного интеграла. **f** — алгебраическое выражение; **x** — переменная интегрирования
int(f, x=a..b)	Вычисление определенного интеграла. **f** — алгебраическое выражение; **x** — переменная интегрирования; **a**, **b** — пределы интегрирования
sum(f,k=m..n)	Суммирование выражения. **f** — алгебраическое выражение; **k** — имя индекса; **m**, **n** — пределы изменения индекса суммирования

Примеры:

> diff(sin(x),x);int(cos(x)+sin(x),x=0..Pi);sum(k^2,k=1..10);

cos(x)

2

385

Под вычислениями понимают и вычисление решений различных уравнений: алгебраических и дифференциальных. Найти решение алгебраического уравнения можно с помощью команды **solve**:

> **solve** (x^2-2=0,x);

$2^{1/2}$, $-2^{1/2}$,

а решение дифференциального уравнения — используя команду **dsolve**:

>**dsolve**(diff(y(x),x$2) - y(x) = sin(x)*x, y(x));

y(x) = - 1/2 cos(x) - 1/2 sin(x) x + _C1 exp(x) + _C2 exp(-x).

Вычислять можно и значение выражений различного типа. В зависимости от типа следует применять свои, присущие этому типу, команды (см. табл. П2.11).

Таблица П2.11

Команды для вычислений

evalb(expr)	Вычисление значения булевского выражения
evalc(expr)	Вычисление значения комплексного выражения
evalf(expr)	Вычисление значения выражения в арифметике чисел с плавающей точкой
evalm(expr)	Вычисление значения матричного выражения

Примеры:

>evalb(x+y=y+x);

true

>evalc(2^(1+I));

2 cos(ln(2)) + 2 I sin(ln(2))

>evalf(2^(1+I));

1.538477803 + 1.277922553 I

> S:=matrix(2,2):T:=matrix(2,2):evalm(S&*T);

$$\begin{bmatrix} S_{1,1}T_{1,1} + S_{1,2}T_{2,1} & S_{1,1}T_{1,2} + S_{1,2}T_{2,2} \\ S_{2,1}T_{1,1} + S_{2,2}T_{2,1} & S_{2,1}T_{1,2} + S_{2,2}T_{2,2} \end{bmatrix}$$

Пакеты

В состав **Maple** входит несколько пакетов **(package)**, то есть специализированных коллекций процедур. В них содержатся дополнительные операции, не включенные в ядро. Чтобы подключить эти пакеты, необходимо использовать команду **with,** написав либо **with(**имя_пакета), например, **with(linalg)** — тогда будут доступны все команды данного пакета, либо **with(**имя_пакета, имя_команды**),** например, **with(plots, odeplot)** — тогда будет доступна только эта команда из пакета. Может оказаться, что некоторые команды пакета совпадают по имени с командами ядра, тогда они переопределяются при подключении пакетов, так, например, команда трассировки **trace** после подключения пакета линейной алгебры **linalg** замещается командой вычисления следа матрицы. Предупреждение о переопределенных командах выдается при загрузке пакета.

Пакет «Линейная алгебра»

Пакет «Линейная алгебра» предназначен для выполнения матричных вычислений.

Основные матричные операции, такие как сложение, умножение матрицы на скаляр, матричное умножение можно выполнять, используя только команды ядра Maple, не прибегая к пакету «Линейная алгебра». Матрицей в **Maple** считается любой двумерный массив, у которого индексы изменяются от единицы до указанной верхней границы. Соответственно вектором считается одномерный массив, начальное значение индекса которого равно единице. Например, квадратную матрицу второго порядка можно описать с помощью команды **>A:=array(1..2,1..2);**. Знаки операций сложения и вычитания матриц совпадают по написанию с обычными (скалярными) операциями, операция умножения записывается с помощью двух символов " &* ". Однако для проведения матричных вычислений привычного для других языков объявления типа объекта не достаточно. Если вы объявите переменные **A:= array(1..2,1..2), B:= array(1..2,1..2),**

C:= array(1..2,1..2) двумерными массивами и напишите **C:=A&*B;** то объект C после выполнения этой команды уже не будет опознан как матрица. Проверить, является ли объект матрицей можно с помощью команды **type(C, matrix)**. Матричные операции в ядре осуществляются только под управлением оператора **evalm**. Напишите команду **C:=evalm(A&*B)**, сравните ее результат с результатом выполнения предыдущей команды и снова задайте вопрос **type(C, matrix)**. Вы увидите, что C опознается вновь как матрица, как и после ее объявления.

Пакет «Линейная алгебра» нужен тогда, когда предполагается проводить сложные матричные вычисления. По своим возможностям он не уступает пакету **Matlab:** также можно искать собственные числа матриц, строить различные разложения, конструировать матрицы специального типа, вычислять функции от матриц. Главное отличие состоит в том, что эти операции проводятся в символьном виде. Мы можем в явном виде построить обратную матрицу, матричную экспоненту, найти, если это возможно, явные зависимости собственных чисел от коэффициентов матрицы.

Сделаем краткий обзор основных команд пакета «Линейная алгебра».

Для того чтобы подключить пакет, необходимо выполнить команду **with(linalg)**.

Команда-описатель **matrix** из библиотеки линейной алгебры имеет вид: **matrix(n,m, [val1, val2, ...])**, здесь **n** — число строк, **m** — число столбцов матрицы, а **val1, val2, ...** — значения элементов матрицы. Элементы матрицы задаются последовательно, строка за строкой, например **[[1,2],[2,1]]**. С точки зрения пакета — эта конструкция является списком списков. Параметры **n** и **m** можно опускать, и тогда размерность матрицы будет определяться списком ее элементов. Существуют и другие формы этого описателя, например: **matrix(n,m,f)**, здесь **f** — функция от индексов матрицы, с помощью которой присваиваются значения соответствующим элементам матрицы. Опишите

переменную как матрицу с помощью команды **matrix** и сравните это описание с описанием, использующим тип **array**. По реакции пакета вы убедитесь, что речь идет об одном и том же объекте, просто синтаксис описателя **matrix** несколько проще.

Вектор можно определить с помощью команды **vector(n, [val1, val2, ...])**, здесь **n** — размерность вектора, а **val1, val2, ...** — значения элементов. Если опустить параметр **n**, то размерность вектора определяется числом явно заданных элементов из списка. Так же как и для матриц, существует другой вариант этого описателя: **vector(n,f)**, здесь **f** — функция, ставящая в соответствие индексу значение компоненты вектора.

Значения элементов матриц и векторов можно задавать как при описании, так и в ходе работы при помощи оператора присваивания. При этом можно определять только часть элементов матрицы или вектора. Увидеть на экране значения введенных коэффициентов матриц и векторов можно только с помощью команд **eval** или **op**, так как матрицы и векторы являются составными объектами.

Приведем несколько примеров задания матриц и векторов:

> A:=matrix (4,3);

$$A := \text{array}(1 .. 4, 1 .. 3, [])$$

> v:=vector([a,b,c]);

$$v := [a, b, c]$$

> L:=matrix (2,2,(i,j)->x^i/y^j);

$$L := \begin{bmatrix} \frac{x}{y} & \frac{x}{y^2} \\ \frac{x^2}{y} & \frac{x^2}{y^2} \end{bmatrix}$$

Для сложения двух матриц (векторов) **A** и **B** одинаковой размерности можно использовать две команды: **matadd(A,B)** — команду пакета «Линейная алгебра» или **evalm(A+B)** — команду ядра. Для команды **matadd** существует расширенный вариант: **matadd(A,B,c,d)** —

в этом случае вычисляется выражение **(c*A+d*B)**, где **c** и **d** — скаляры.

Умножить матрицу A на матрицу B также можно двумя способами: **multiply (A,B)** и **evalm(A&*B)**. Проиллюстрируем это примерами:

>F:=matrix(2,2,[1,2,3,4]);G:=matrix(2,2,[4,3,2,1]);T:=matadd(F, G,m,n);M:=evalm(F&*G); evalm(F^3);

$$F := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$G := \begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix}$$

$$T := \begin{bmatrix} m + 4n & 2m + 3n \\ 3m + 2n & 4m + n \end{bmatrix}$$

$$M := \begin{bmatrix} 8 & 5 \\ 13 & 20 \end{bmatrix}$$

$$\begin{bmatrix} 37 & 54 \\ 81 & 118 \end{bmatrix}$$

В Maple также существуют функции, традиционно использующиеся при матричных вычислениях (см. табл. П2.12).

Таблица П2.12

Функции матричных вычислений

cond(A)	Вычисление числа обусловленности
det(A)	Вычисление определителя матрицы
inverse(A)	Вычисление обратной матрицы
rank(A)	Вычисление ранга матрицы
trace(A)	Вычисление следа матрицы

Пример использования функций:

> A:=matrix(2,2,[-1,2,0,3]);cond(A);det(A);rank(A);inverse(A);

$$A := \begin{bmatrix} -1 & 2 \\ 0 & 3 \end{bmatrix}$$

$$5$$

$$-3$$

$$2$$

$$\begin{bmatrix} -1 & \frac{2}{3} \\ 0 & \frac{1}{3} \end{bmatrix}$$

В заключение приведем пример символьных вычислений с матрицами, а именно вычислим обратную к заданной матрице, ее собственные числа и матричную экспоненту:

> A:=matrix(2,2,[[a,b],[c,d]]);

$$A := \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

> inverse(A);

$$\begin{bmatrix} \frac{d}{ad - bc} & -\frac{b}{ad - bc} \\ -\frac{c}{ad - bc} & \frac{a}{ad - bc} \end{bmatrix}$$

> eigenvalues(A);

$$1/2\ a + 1/2\ d + 1/2\ (a^2 - 2\ a\ d + d^2 + 4\ b\ c)^{1/2},$$

$$1/2\ a + 1/2\ d - 1/2\ (a^2 - 2\ a\ d + d^2 + 4\ b\ c)^{1/2}$$

>exponential(A);

Последний результат мы не приводим ради экономии места. Советуем получить его самостоятельно.

Пакет «Визуализация решений дифференциальных уравнений»

В Maple основной формой представления решения дифференциального уравнения считается аналитическая форма записи. Численное решение строят, как правило, если не удается найти решение в замкнутой форме. Аналитическое и численное решения, естественно, различаются и формой представления — в первом случае речь идет об алгебраическом выражении, во втором — о списке, содержащем дискретные значения аргумента и функции, удовлетворяющие уравнению. Если решение можно построить в замкнутой форме, то, как мы увидим в дальнейшем, его достаточно просто и визуализировать. Сложнее дело обстоит с таблицами. Построить график функции, заданной таблично, не просто. Для облегчения этой работы и служат процедуры пакета «Визуализация решений дифференциальных уравнений». Точнее речь идет о двух пакетах — **plots** и **Detools**, которые мы условно объединили под общим названием. Существующие в них процедуры позволяют строить графики решений дифференциальных уравнений, найденных с помощью уже упоминавшейся команды ядра **dsolve**, и фазовые портреты.

Начнем с команды **dsolve**, принадлежащей ядру, решающей не только дифференциальные уравнения, но и системы дифференциальных уравнений.

Сначала научимся задавать уравнение, так как этого требует **Maple**. Описать дифференциальное уравнение, не используя обозначение производной искомой функции, невозможно. Напомним, что производная от любого алгебраического выражения вычисляется с помощью команды: **diff(a, x1, x2, ..., xn)**, где **a** — алгебраическое выражение; **x1, x2, ..., xn** — переменные, по которым берется производная. Наша задача более простая — нам надо указать пакету, что в дифференциальное уравнение входит первая или более высокая производная искомой функции. Для этих целей удобно использовать оператор **$**, позволяющий упростить описание последовательности выражений. Например, для вычисления 4-ой производной функции f(x)

можно написать команду **diff(f(x),x,x,x,x)**, проще это сделать так: **diff(f(x),x$4)**. Пусть нужно решить уравнение $y''_{xx}(x) - y(x) = x\sin x$. Описание уравнение с помощью команды **diff** выглядит следующим образом:

>dsolve(diff(y(x),x$2) - y(x) = sin(x)*x, y(x));

После запятой, заканчивающей описание уравнения, указывается искомая функция. В результате выполнения команды получим следующее решение:

y(x) = – 1/2 cos(x) – 1/2 sin(x) x + _C1 exp(x) + _C2 exp(-x),

где _C1 и _C2 обозначают неопределенные константы.

Если нам заданы граничные или начальные условия, и мы хотим найти решение, удовлетворяющее этим условиям, то нам необходимо сформировать множество, включающее и уравнение, и условия. Например, для уравнения $y''_{xx} = x$ с начальными условиями: y(0) = 1; y(1) = 1 оператор **dsolve** следует записать так:

> dsolve({diff(y(x),x$2) = x, y(1)=0, y(0)=1}, y(x));

Когда уравнение нельзя решить аналитически, можно попытаться найти численное решение. Команда **dsolve** в этом случае будет иметь более сложный синтаксис: **dsolve(deqns, vars, eqns)**, где **deqns** — описание решаемых уравнений и начальных условий, **vars** — неизвестные функции, **eqns** — дополнительные опции. Нам понадобится только несколько основных опций. Начнем с опции **type**. По умолчанию ее значение равно **exact**, и тогда Maple ищет аналитическое решение. Если ей придать значение **numeric**, то будет строиться численное решение. Можно явно указать численный метод, с помощью которого строится решение. За выбор метода отвечает опция **method**. Возможные значения этой опции приведены в табл. П2.13.

Приведенные в табл. П2.13 методы имеют большое число различных «настроек» или параметров, таких как задаваемая относительная и абсолютная погрешности решения, допустимое на шаге интегрирования число вычислений правых частей и других, необходимых для получения правильного численного решения конкретной за-

дачи. Авторы пакета обеспечили доступ ко всем этим параметрам за счет усложнения синтаксиса вызова методов (см. соответствующие разделы **Help**). Те, кто знаком с программными реализациями перечисленных методов на Фортране, легко справятся с задачей настройки метода без подсказок.

Таблица П2.13

Значения опции method

rkf45	Метод Рунге-Кутта 4–5 порядков с автоматическим выбором шага (метод 4-го порядка — маршевый, метод 5-го порядка используется для контроля погрешности и выбора величины шага)
dverk78	Метод Рунге-Кутта 7–8-го порядков с автоматическим выбором шага
classical	«Классические методы». Содержит несколько методов (их выбор управляется дополнительной опцией), таких как явный и неявный метод Эйлера, метод трапеций, методы Рунге–Кутта 3-го порядка и др., работающие с фиксированным шагом
gear	Экстраполяционный метод, построенный на методе Гира
mgear	Программа Гира для решения как жестких, так и нежестких уравнений
lsode	Программа из пакета ODEPACK для решения как жестких, так и нежестких уравнений.

Поясним на примерах, как пользоваться командой **dsolve**. Построим решение уравнения $y'_x = x^2 \sin x$ аналитически и двумя численными методами. Результаты вычислений будем сравнивать в заданной точке, например, **x=7**. Далее через **F1** обозначено аналитическое решение; через **F2** — решение, полученное с помощью метода Рунге–Кутта 4–5; а через **F3** — решение, полученное с помощью метода Рунге–Кутта 7–8 (более точное).

```
> F1:=op(2,dsolve({diff(y(x),x)=x*x*sin(x),y(1)=0}, y(x)));
> F2:=dsolve({diff(y(x),x)=x*x*sin(x),y(1)=0},
y(x),type=numeric,method=rkf45);
```

> F3:=dsolve({diff(y(x),x)=x*x*sin(x),y(1)=0},
y(x),type=numeric,method=dverk78);

> evalf(subs(x=7,F1));

> op(2,F2(7));

> op(2,F3(7));

$$F1 := -x^2\cos(x) + 2\cos(x) + 2\sin(x)\ x - \cos(1) - 2\sin(1)$$

F2 := proc(rkf45_x) ... end

F3 := proc(dverk78_t) ... end

-28.45883785

y(x) = -28.45883773990384

y(x) = -28.45883784598292

Обратите внимание на то, как записано выражение для вычисления F1. Имени F1 присваивается второй операнд выражения, выдаваемого в качестве результата командой dsolve. Дело в том, что команда dsolve

> dsolve(diff(y(x),x$2) - y(x) = sin(x)*x, y(x));

выдает результат в виде выражения:

y(x) = - 1/2 cos(x) - 1/2 sin(x) x + _C1 exp(x) + _C2 exp(-x),

которое можно трактовать как бинарную операцию " = ", имеющую два операнда: левую часть y(x) и правую часть (- 1/2 cos(x) - 1/2 sin(x) x + _C1 exp(x) + _C2 exp(-x)). Правая часть представляет собой обычное выражение, значение которого можно вычислять в заданной точке x. Делается это с помощью подстановки **subs(x=7,F1);** и последующего вычисления численного значения **evalf(subs(x=7,F1))**. Как уже отмечалось, численное решение выдается в виде списка, у которого в первой колонке записывается значение аргумента, а во второй — значение искомой функции. Команда **op(2,F2(7))**, таким образом, нужна для того, чтобы вывести только значение функции. При этом содержимое второй колонки таблицы равно выражению y(x) = -28.45883773990384. Если бы мы хотели построить график такой табличной функции, нам бы пришлось снова извлекать из ответа

только второй операнд. Чтобы лучше понять написанное пояснение, разберите ниже приведенный пример.

Пример. Другой способ решения той же задачи.

Решаем уравнение аналитически:

> F1:=dsolve({diff(y(x),x)=x*x*sin(x),y(1)=0},y(x));

$$F1 := y(x) = -x^2\cos(x) + 2\cos(x) + 2x\sin(x) - \cos(1) - 2\sin(1)$$

Анализируем тип полученного выражения:

> whattype(F1);

=

Выясняем, что это выражение типа «равенство», и выделяем второй операнд:

> f1:=op(2,F1);

$$f1 := -x^2\cos(x) + 2\cos(x) + 2x\sin(x) - \cos(1) - 2\sin(1)$$

Либо, что то же самое, просим выделить правую часть выражения:

> f2:=rhs(F1);

$$f2 := -x^2\cos(x) + 2\cos(x) + 2x\sin(x) - \cos(1) - 2\sin(1)$$

Используя локальную подстановку, вычисляем значение в нужной точке

> evalf(subs(x=7,f2));

-28.45883785

и переходим к построению численного решения методом Рунге–Кутта с автоматическим выбором шага:

> F2:=dsolve({diff(y(x),x)=x*x*sin(x),y(1)=0},y(x),type=numeric,method=rkf45);

F2 := proc(rkf45_x) ... end

В этом ответе нет ничего загадочного, просто пакет выполнил всю предварительную работу и теперь ждет от вас указания, в какой точке вычислить решение.

> F2(7);

[x = 7, y(x) = -28.45883773990384]

Убедимся в том, что выведенное выражение принадлежит к типу «список»:

> whattype(F2(7));

list

Выделим из списка элемент, содержащий значение численного решения в указанной точке:

> F2(7)[2];

y(x) = -28.45883773990384

Это же можно сделать и по-другому:

> op(2,F2(7));

y(x) = -28.45883773990384

> rhs(F2(7)[2]);

-28.45883773990384

Теперь сменим метод решения:

>F3:=dsolve({diff(y(x),x)=x*x*sin(x),y(1)=0},y(x),type=numeric, method=dverk78);

F3 := proc(dverk78_t) ... end

> rhs(F3(7)[2]);

-28.45883784598292

Как мы видим, добраться до нужного результата даже в этом простом примере достаточно сложно, именно поэтому мы советуем пользоваться процедурами пакета «Визуализация решений дифференциальных уравнений», которые делают за вас всю «черную» работу.

Теперь перейдем к решению систем дифференциальных уравнений. Нам понадобится все та же команда **dsolve**, но теперь вместо од-

ного уравнения необходимо задать множество уравнений и начальные условия для них.

Пусть требуется найти аналитическое решение системы уравнений

$$\begin{cases} y'_x = x + z(x) \\ z'_x = 1 - y(x) \end{cases}$$

с начальными условиями y(1) = 0; z(0) = 1 в точке $x = 7$. Тогда мы должны записать и выполнить следующие операторы:

> F5:=dsolve({diff(y(x),x)=x+z(x),diff(z(x),x)=1-y(x),y(1)=0,z(0)=1}, {y(x),z(x)});

> evalf(subs(x=7,F5));

$$F5 := \{y(x) = -\frac{\cos(x)(\sin(1)+2)}{\cos(1)} + \sin(x) + 2; z(x) = \frac{\sin(x)(\sin(1)+2)}{\cos(1)} + \cos(x) - x\}$$

{y(7) = -1.307814879, z(7) = -2.790979495}

Они отличаются от приведенных ранее только тем, что в множество (первый аргумент процедуры **dsolve**) включено описание не одного уравнения, а двух. Аналогичные изменения претерпел и второй аргумент — теперь это множество, содержащее две искомые функции**{y(x), z(x)}**. Обратите внимание, что в данном случае мы не стали выделять в ответе второй операнд результата и подставили значение x = 7 как в левую, так и в правую части одновременно.

С командой **dsolve** связаны несколько процедур, облегчающих визуализацию решений. Прежде всего, это команда **odeplot** из пакета **plots**. Рассмотрим пример. Решим численно уравнение y' = y, y(0) = 1 и построим график решения на промежутке [–1,1]:

>with(plots):

> p:= dsolve({D(y)(x) = y(x), y(0)=1}, y(x),type=numeric):

> odeplot(p,[x,y(x)],-1..1):

В приведенном фрагменте есть две особенности. Во-первых, производная искомой функции записана в виде D(y)(x), что является допустимой конструкцией (см. описание оператора дифференцирования в **Help**). Во-вторых, в качестве первого параметра оператора

odeplot подставлено имя p, что делает запись более наглядной. Второй параметр команды **odeplot** представляет из себя список, содержащий аргумент и искомую функцию, чей график мы строим. Последний параметр легко узнаваем — это конструкция типа «диапазон». В качестве последнего примера приведем фрагмент, где ищется решение системы. Этот фрагмент интересен тем, что символические имена даны как самой системе — sys, так и искомым функциям — fcns.

```
> sys := diff(y(x),x)=z(x),diff(z(x),x)=y(x):  fcns := {y(x), z(x)}:
> p:= dsolve({sys,y(0)=0,z(0)=1},fcns,type=numeric):
> odeplot(p, [x,y(x)], -4..4, numpoints=25):
```

Для построения графиков решений дифференциальных уравнений можно также воспользоваться командой **DEplot** из пакета **Detools**. У этой команды больше возможностей, чем у команды **odeplot**. Заинтересованных читателей отсылаем к соответствующему разделу Help. Мы же рассмотрим процедуру, позволяющую строить фазовые портреты.

Рассмотрим пример. Процедура **phaseportrait** требует описания решения дифференциального уравнения, в данном случае уравнения третьего порядка, соответствующих начальных условий и диапазонов изменения координат на графиках. Косая черта, использованная в примере, является символом переноса строки:

```
with(DEtools):
> phaseportrait(cos(x)*diff(y(x),x$3)-
diff(y(x),x$2)+Pi*diff(y(x),x)=y(x)-x,\
>y(x),x=-2.5..1.4,[[y(0)=1,D(y)(0)=2,(D@@2)(y)(0)=1]],y=-4..5);
```

Пусть вас не смущает, что в результате будет построен график зависимости y = y(x) — это частный случай фазового портрета. Более сложным является пример построения фазового портрета для системы из трех уравнений:

```
> phaseportrait([D(x)(t)=y(t)-z(t),D(y)(t)=z(t)-x(t),D(z)(t)=x(t)-
y(t)*2], \
```

```
> [x(t),y(t),z(t)],t=-2..2,[[x(0)=1,y(0)=0,z(0)=2]],stepsize=.05, \
>
scene=[z(t),x(t)],linecolour=sin(t*Pi/2),method=classical[foreuler]);
```

Графика

Таблица П2.14

Опции двумерной графики

Вид осей координат	**axes** = **a**	**a** принимает значения: **boxed** (оси в виде прямоугольника), **normal** (оси с началом координат в центре рисунка), **frame** (оси с началом координат в левом нижнем углу рисунка), **none** (вывод графика без осей)
Цвет графика	**color = c**	**c** — один из ниже перечисленных цветов: black — черный blue — синий navy — темно-синий cyan — цианový brown — коричневый gold — золотой green — зеленый gray — серый **red** — **красный** и др.
Стиль отображения графика	**style** = **s**	**s** — либо **line** (линия), либо **point** (точки)
Толщина линии	**thickness = n**	**n** принимает значения **1**, 2,3, ... Эта величина влияет на график только, если **style=line**
Заголовок графика	**title = s**	**s** — любая строка, заключенная в кавычки, например, *`График точного решения`*, или строковая переменная

Графики являются самым наглядным способом представления полученных результатов. Исходные данные для построения графиков могут быть заданы в виде функций, массивов, последовательностей,

графических структур. В качестве параметров в командах вывода изображения на экран указываются: исходные данные, желаемый цвет графика, масштаб, интервалы изменения значений, вид графика и т. д. Многие параметры можно опускать, и **Maple** сам подставит значения по умолчанию.

В **Maple** существуют команды как двумерной, так и трехмерной графики. Наиболее универсальные команды расположены в ядре. Для использования всей графической мощи **Maple** нужно подключить соответствующую библиотеку с помощью команды **with(plots)**.

Во многих командах используются одни и те же опции. Опции располагаются в командах сразу за обязательными параметрами в произвольном порядке. Все они имеют свои значения по умолчанию (в табл. П2.14 они подчеркнуты). В табл. П2.14 перечислены лишь самые распространенные опции.

Команды двумерной графики

Самая простая команда — это команда **plot** — построение графика функции, зависящей от одной переменной. Команда имеет следующий формат:

plot(f(x), x=a..b, opt),

где **f(x)** — вещественная функция; зависящая только от одной переменной **x**; **a..b** — интервал изменения аргумента; **opt** — опции графика.

Можно указывать и границы изменения аргумента, и границы по оси ординат: **plot(f(x), x=a..b, y=c..d, opt)**. В одном окне можно отобразить несколько графиков. Тогда в качестве параметров нужно задать не функцию, а множество функций: **plot({f1(x),f2(x),...},x=a..b,y=c..d,opt)**. Для построения графика функции, заданной параметрически, необходимо использовать команду **plot** в соответствии со следующим форматом:

plot([f1(t), f2(t), t=a..b]),

где f1(t), f2(t) — функции координат, зависящие от параметра; a..b — интервал изменения параметра.

Если функция в команде **plot** является табличной, то следует сформировать список соответствующих пар — абсцисс и ординат: **plot([x1, y1, x2, y2, …], opt)** или **plot([x1,y1],[x2, y2], opt)**, где **opt** — опции.

В пакете имеются также команды отображения n-угольника, отображения графиков в логарифмическом масштабе, построение векторного поля, анимация и многие другие.

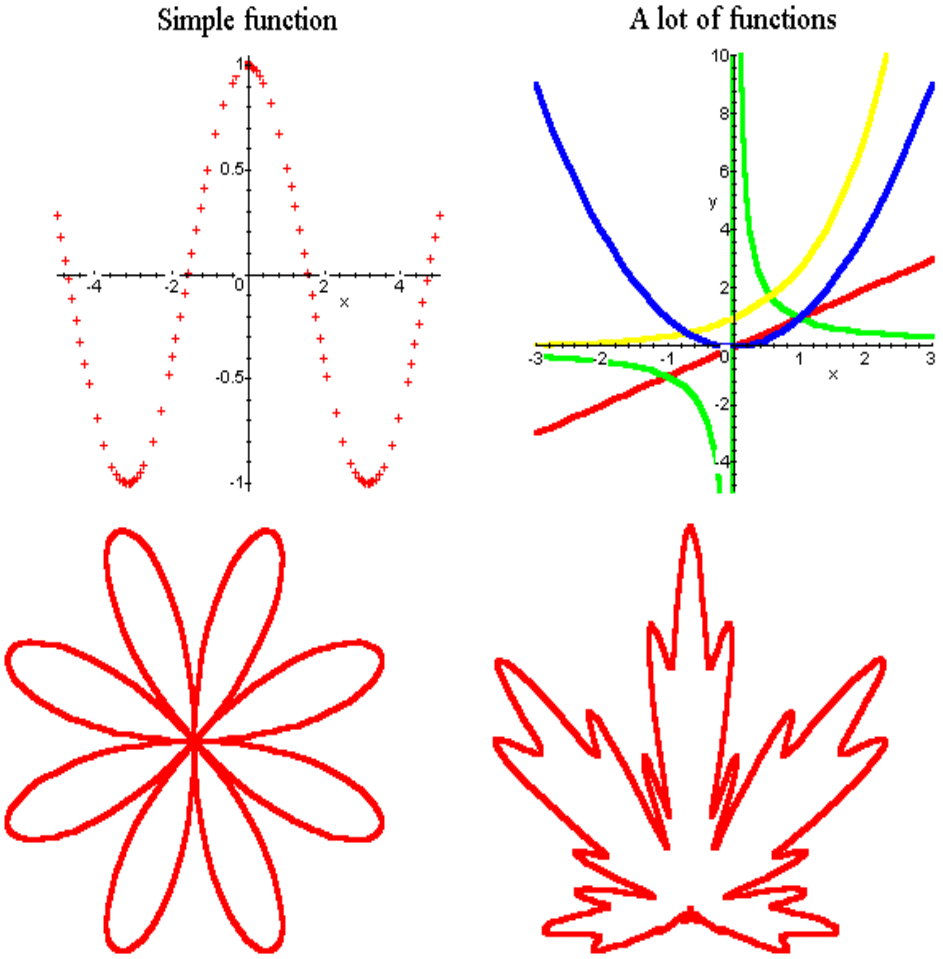


Рис. П2.1. Графики, построенные с помощью команды plot

Ниже приведены примеры использования команды **plot**, а на рис. П2.1 — построенные с ее помощью графики.

> plot(cos(x), x=-5..5,title=`Simple function`,style=POINT);

> plot({x,x^2,1/x,exp(x)},x=-3..3,y=-5..10,title =`A lot of functions`,thickness=3);

>

plot([sin(4*x),x,x=0..2*Pi],coords=polar,thickness=3,axes=none);

> r(t):=100/(100+(t-Pi/2)^8)*(2-sin(7*t)-cos(30*t)/2):

> plot([r(t),t,t=-Pi/2..3/2*Pi],coords=polar,axes=none,thickness=3);

Двумерные графические структуры

Перечислим основные графические структуры (двумерные графические примитивы), которые умеет рисовать пакет (табл. П2.15).

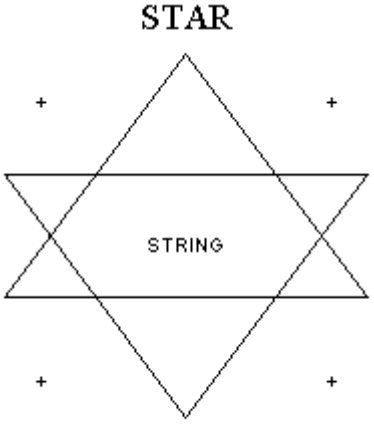
Таблица П2.15

Основные графические структуры

points([x1,y1],[x2,y2],...[xn,yn])	Набор точек с координатами **[x1,y1],[x2,y2],...**
curves([[x11,y11],...[x1n,y1n]], [[x21,y21],...[x2n,y2n]],... [[xm1,ym1],...[xmn,ymn]])	Последовательность из **m** незамкнутых ломаных по **n** точек в каждой. Точки ломаных будут последовательно соединены отрезками прямых
polygons([[x11,y11],...[x1n,y1n]], [[x21,y21],...[x2n,y2n]],... [[xm1,ym1],...[xmn,ymn]])	Последовательность из **m** замкнутых ломаных по **n** точек в каждой. Точки будут соединены отрезками прямых, причем последняя будет соединена с первой
text([x,y],`string`,horizontal, vertical)	Вывод текста в графическом окне. **string** — выводимая строка; **x,y** — координаты начала вывода текста; **horizontal** — вид выравнивания по горизонтали: **alignleft** (по левому краю) или **alignright** (по правому краю); **vertical** — вид выравнивания по горизонтали: **alignabove** (по верхнему краю) или **alignbelow** (по нижнему краю)
title(string)	Отображение заголовка рисунка; **string** — произвольная строка символов
axesstyle(type)	Вид отображения осей координат; принимает значения **box, frame, normal, none** (см. опцию **axes** команды **plot**)

Для отображения графических структур **s1, ..,sn** на экран следует использовать команду **plot ({s1,..,sn})**. Пример, иллюстрирующий использование графических структур пакета **Maple**:

```
> a1:=POLYGONS([[1,1],[1.5,0],[0.5,0]],[[1,-
0.5],[1.5,0.5],[0.5,0.5]]):
> a2:=POINTS([1.4,0.8],[0.6,0.8],[1.4,-0.35],[0.6,-0.35]):
> a3:=TEXT([1,0.2],`STRING`):
> PLOT(a1,a2,a3,TITLE(`STAR`),AXESSTYLE(NONE));
```



Программирование

Как уже отмечалось, **Maple** обладает собственными алгоритмическими конструкциями, позволяющими писать программы.

***Условный оператор*.** В **Maple** условный оператор имеет вид:

if cond then expr1 else expr2 fi;

где **cond** — логическое (булевское) выражение; **expr1** — оператор, выполняющийся, если условие истинно; **expr2** — оператор, выполняющийся, если условие ложно.

Заметим, что можно использовать сокращенную форму условного оператора: **if cond then expr fi;**. Если нужно осуществить сложную проверку, тогда следует применить наиболее полную форму условного оператора:

```
if cond1 then expr1
    elif cond 2 then expr2 ...
        elif cond n then exprn
```

else expr0

fi;

Примеры использования условного оператора:

> lambda:=1: x:=4:

> if lambda>0 then print(`lambda>0`); else print(`lambda<0`); fi;

> if x<0 then a:=1 elif (x>=0)and(x<10) then a:=2 else a:=3; fi;

lambda>0

a := 2

Операторы цикла. В **Maple** существует несколько видов циклов. Во всех случаях тело цикла заключено между операторными скобками **do** и **od**. Начнем с самого простого:

for var from val1 by val2 to val3 do expr od;

где **var** — переменная, которая является счетчиком цикла; **val1** — начальное значение переменной; **val2** — шаг приращения; **val3** — конечное значение переменной; **expr** — тело цикла (любые операторы **Maple**).

Пример:

> for i from 3 by 2 to 7 do 2*i; od;

6

10

14

Другой тип цикла («пока») имеет вид:

while cond do expr od;

Тело цикла исполняется, пока значение выражения expr истинно (**true**). Как только значение expr становится ложным (**false**), происходит выход из цикла.

Пример:

> a:=1: while a<3 do b:=2/a; a:=a+1; od;

b := 2

a := 2

b := 1

a := 3

Следующий тип цикла является объединением двух предыдущих:

for var from val1 by val2 to val3 while cond do expr od;

Тело цикла выполняется, пока cond истинно, и переменная var (начальное значение — **val1**, приращение — **val2**) не стала больше **val3**. На следующем примере видно, что переменная i должна меняться от 1 до 15 с шагом 2, но при достижении i значения 7 происходит выход из цикла:

```
> for i from 1 by 2 to 15 while i<=5 do i;  od;
                                1
                                3
                                5
```

Последняя модификация цикла ориентирована на работу с символьными выражениями, в частности, с последовательностями. Вот соответствующая форма цикла:

for var in expr1 do expr2 od;

Переменная **var** последовательно принимает значения выражения **expr1**. Выражение **expr2** является телом цикла. Следующий пример демонстрирует использование данного типа цикла. В нем мы суммируем значения последовательности:

```
> sum1:=0: seq1:=(1,2,3,4): for z in seq1
> do sum1:=sum1+z;  od;
                          sum1 := 1
                          sum1 := 3
                          sum1 := 6
                          sum1 := 10
```

При написании собственных программ бывает удобно пользоваться процедурами-функциями и процедурами. Один из способов задания процедуры-функции предполагает использование символа ->.

Команда имеет следующий вид:

Имя_функции:= (var1,var2,…)->expr; где **var1,var2,….** — имена формальных параметров, **expr** — выражение, реализующее те-

ло процедуры-функции. При выполнении учебного задания, процедуры-функции можно использовать, например, при формировании коэффициентов матриц:

fun:=(i,j)->x^i/y^j; B:= matrix(2,2,fun);

Об использовании процедур см. соответствующий раздел Help.

Пакет «LinearAlgebra» (Maple 7)

В версии Maple 7 существуют уже два пакета для решения задач линейной алгебры — новый «LinearAlgebra» и ранее рассмотренный нами «linalg».

Пакет «LinearAlgebra» уже обладает новыми матричными типами — Matrix и Vector, матричные операции над которыми можно записывать в привычной математической нотации, и позволяет использовать численные алгоритмы из библиотеки NAG для работы с разреженными матрицами.

Авторы рекомендуют использовать пакет «linalg» для решения теоретических задач, а пакет «LinearAlgebra» — для практических, связанных с численным решением больших, разреженных систем уравнений.

Рассмотрим возможности пакета «LinearAlgebra». Начнем с описания матриц и векторов. В пакете «LinearAlgebra» матрицы и векторы создаются на основе структуры данных «rtables», которую авторы относят к низкоуровневым структурам. На основе этой структуры могут быть построены любые матричные объекты, но это требует достаточно глубоких знаний о структуре пакета. Основными же являются два оператора «высокого уровня» Matrix и Vector, задающие матрицы и векторы в более простом для пользователя виде (в пакете «linalg» аналогом низкоуровневой структуры является структура «array», а высокоуровневых операторов — операторы matrix и vector).

Матрицы

Матрица (Matrix) представляется с помощью типа данных **rtable** с указанием подтипа **Matrix**. Матрицы создаются с помощью конструктора **Matrix(..)**, или с помощью упрощенного конструктора вида (<a,b,c>), в данном случае это — пример задания вектора-столбца с тремя элементами a,b,c.

Матрицы могут быть **прямоугольными, треугольными, хессенберговыми, ленточными и диагональными (rectangular, triangular**, **Hessenberg**, **band**, **diagonal)**. Каждый из типов может быть представлен как полностью заполненная матрица, так и разреженная. Среди конкретных типов могут быть подтипы:

1) **rectangular**: симметрическая, кососимметрическая, эрмитовая, косоэрмитовая матрица (symmetric, skew-symmetric, hermitian, skew-hermitian);

2) **triangular**: верхняя треугольная, единичная в виде верхней треугольной матрица (upper, upper unit); нижняя треугольная, единичная в виде нижней треугольной матрица (lower, lower unit);

3) **Hessenberg**: верхняя, нижняя матрица (upper, lower).

Существуют конструкторы специальных матриц — единичной, нулевой и матрицы-константы (Identity Matrix, the Zero Matrix, and the Constant Matrix). Ключевое слово Matrix может быть использовано для проверки типа объекта (type(A,Matrix)). Элементы матрицы доступны с помощью оператора **A[i,j] с привычным синтаксисом.** Матричные операции **A+B**, **A-B**, и **A . B** выполняются над матрицами непосредственно, как того требуют определения матричных операторов. Групповые операции над элементами матриц осуществляются с помощью команд **map**, **Map**, **map2**, и **Map2**.

Векторы

Тип **Vector** создается с помощью типа **rtable** с указанием подтипов **Vector[column]** (вектор-столбец) или **Vector[row] (вектор-строка)**. Существует конструктор для создания векторов Vector(..). Существуют конструкторы специальных типов векторов — орта, ну-

левого вектора и вектор-константы (Unit Vector, the Zero Vector, и Constant Vector). Ключевое слово Vector может быть использовано в операторах для определения типа объекта. Оператор **V[i]** дает доступ к элементу вектора. Групповые операции над векторами осуществляются с помощью операторов **map**, **Map**, **map2** и **Map2**.

Полная синтаксическая форма конструктора матриц имеет вид:

Matrix(r, c, init, ro, sc, sh, st, o, dt, f, a),

где

r (может быть опущен) — строчная размерность матрицы — неотрицательное целое или диапазон с нижним значением 1;

c (может быть опущен) — столбцевая размерность матрицы — неотрицательное целое или диапазон с нижним значением 1;

init (может быть опущен) — любой из операторов procedure, table, array, list, Array, Matrix, Vector, набор уравнений, выражение алгебраического типа (algebraic) для задания начальных значений матрицы;

ro (может быть опущен) — BooleanOpt(readonly); булевская константа, указывающая на то, что элементы матрицы доступны или только для чтения, или для записи тоже;

sc (может быть опущен) — уравнение вида **scan=name** или **scan=list,** указывающее, как интерпретировать список начальных значений для матрицы,

sh (может быть опущен) — уравнение вида **shape=name** или **shape=list,** указывающее на вид функции индексирования;

st (может быть опущен) — уравнение вида **storage=name**, указывающее на внутреннюю форму представления матрицы;

o (может быть опущен) — уравнение вида **order=name**, где name — это либо **C_order,** либо **Fortran_order указывающее на то, как данные располагаются внутри двумерного массива**;

dt (может быть опущен) — уравнение вида **datatype=name**, где name — любой из типов, complex[8], или integer[n] for n=1,2,4, или 8; для указания типа коэффициентов;

f (может быть опущен) — уравнение вида **fill=value**, где значение value используется для заполнения элементов матрицы с типом, совпадающим с указанным в параметре dt;

a (может быть опущен) — уравнение вида **attributes=list**, где list определяет дополнительные математические свойства матрицы.

Примеры использования

Оператор **Matrix(r)** создает квадратную **r x r** матрицу, значения которой определяются значением параметра **f** (**fill** — заполнитель, по умолчанию 0). Если строчная размерность не задана, она автоматически полагается равной нулю, и создается так называемая матрица с нулевой размерностью. Оператор **Matrix(r,c)** создает прямоугольную **r x c** матрицу.

Оператор **Matrix(r,c,init) или Matrix(init)** создает матрицу, форма и значения которой определяются параметром **init**. В качестве значений параметра **init** допустимы только следующие формы.

procedure. В качестве входных значений процедура должна принимать индексы, а возвращать значения элементов.

expression of type algebraic. Выражение интерпретируется как функция двух аргументов — индексов, возвращающая значение элемента матрицы:

```
Matrix(4,(i,j)->i+j);
```

table. Для инициализации может быть использована таблица подходящих размеров:

```
> T:=table();
> T[1,1]:=1;T[4,4]:=2;
> Matrix(4,T);
```

set of equations. Можно задать множество уравнений вида **(i, j) = <value>** для задания элементов матрицы:

```
> b:=Matrix(2,2,{(1,1)=1,(1,2)=2});
```

array or Array, ***Matrix list, or list of lists***.

Для инициализации матрицы может быть использован любой массив, полученный на основе типа *table* или *rtable*, любая матрица или список списков.

Оператор **Matrix(r, shape=identity)** создает **r x r** единичную матрицу. Параметр задает форму матрицы. В качестве имен функции индексирования, определяющей форму матрицы, можно использовать: antisymmetric, hermitian, skewhermitian, antihermitian, triangular, Hessenberg, identity, scalar[x], zero, constant[x], diagonal, band[b], band[b1, b2], symmetric, skewsymmetric.

Оператор **Matrix(r, shape=identity, storage=** triangular[upper]**)** создает **r x r** единичную матрицу, которая хранится в памяти в специальной форме указанного типа, то есть в форме верхней треугольной матрицы. Речь в данном случае идет о физическом расположении матрицы в памяти. То есть в памяти выделено место только для верхнего треугольника квадратной матрицы. В качестве других форм хранения можно использовать: rectangular, triangular[upper], triangular[lower], triangular[upper, strict], triangular[lower, strict], Hessenberg[upper], Hessenberg[lower], band[b1, b2], band[b], diagonal, empty, sparse, sparse[upper], sparse[lower].

Оператор **Matrix(r, shape=** triangular**, order= Fortran_order)** создает квадратную **r x r** матрицу, элементы которой располагаются по столбцам.

Оператор **Matrix(r, shape=** triangular**, order= Fortran_order, datatype= float[8])** создает **r x r** матрицу, элементы которой располагаются по столбцам и являются вещественными числами (аппаратными).

Оператор **Matrix(r, shape=** triangular**, order= Fortran_order, datatype= float[8], fill=1.0)** создает **r x r** матрицу, элементы которой располагаются по столбцам и являются вещественными числами, равными 1,0.

Полной синтаксической конструкцией для создания векторов является конструкция

Vector[o](d, init, ro, sh, st, dt, f, a, o),

где

[o] (может быть опущен) — тип вектора, может принимать значения [row] (строка) или [column] (столбец);

d (может быть опущен) — размерность вектора, положительное целое или целый интервал, с нижним значением 1;

init (может быть опущен) — любой из операторов procedure, table, array, list, Array, Matrix, Vector, набор уравнений, выражение алгебраического типа (algebraic) для задания начальных значений;

ro (может быть опущен) — BooleanOpt(readonly); булевская константа, указывающая на то, что элементы матрицы доступны только для чтения, или для записи тоже;

sh (может быть опущен) — уравнение в виде **shape=name** or **shape=list**, указывающее возможные функции индексирования для вектора (indexing functions);

st (может быть опущен) — уравнение в виде **storage=name**, где name — один из допустимых способов хранения вектора в памяти;

dt (может быть опущен) — уравнение вида **datatype=name**, где name — любой из типов, complex[8], или integer[n] for n=1,2,4, или 8; для указания типа коэффициентов;

f (может быть опущен) — уравнение вида **fill=value**, где значение value используется для заполнения элементов матрицы с типом, совпадающим с указанным в параметре dt;

a (может быть опущен) уравнение вида **attributes=list**, где list определяет дополнительные математические свойства матрицы;

o (может быть опущен) уравнение вида form **orientation=name**, где **name** — row или column для создания векторов-столбцов и векторов-строк.

Все опции аналогичны соответствующим для матрицы, за исключением **shape=name** или **shape=list**.

К числу встроенных функций индексирования для векторов относятся только: **constant, scalar ,unit , zero**.

Для более простого создания векторов и матриц вводятся конструкторы.

- < a, b, c > для задания объектов построчно;
- < a | b | c > для задания объектов по столбцам.

Примеры.

Построение вектора-столбца:

> **< 1, 2, 3 >;**

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Построение вектора-строки:

> **< 1 | 2 | 3 >;**

$$[1, 2, 3]$$

Построение матрицы:

> **< < 1, 2, 3 > | < 4, 5, 6 > >;**

$$\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

Операторы пакета «LinearAlgebra»

Пакет содержит следующие команды: Add; Adjoint; BackwardSubstitute; BandMatrix; Basis; BezoutMatrix; BidiagonalForm; BilinearForm; CharacteristicMatrix; CharacteristicPolynomial; Column; ColumnDimension; ColumnOperation; ColumnSpace; CompanionMatrix; ConditionNumber; ConstantMatrix; CreatePermutation; CrossProduct; DeleteColumn; DeleteRow; Determinant; DiagonalMatrix; Dimension; Dimensions; DotProduct; Eigenvalues; Eigenvectors; Equal; ForwardSubstitute; FrobeniusForm; GaussianElimination; GenerateEquations; GenerateMatrix; GivensRotationMatrix; GramSchmidt; HankelMatrix; HermiteForm; HermitianTranspose; HessenbergForm; HilbertMatrix; HouseholderMatrix; IdentityMatrix; IntersectionBasis; IsDefinite; IsOrthogonal; IsSimilar; IsUnitary; JordanBlockMatrix; JordanForm; LeastSquares;

LinearSolve; LinearSolveItr; LUDecomposition; Map; Map2; MatrixAdd; MatrixInverse; MatrixMatrixMultiply; MatrixNorm; MatrixScalarMultiply; MatrixVectorMultiply; MinimalPolynomial; Minor; Multiply; Norm; Normalize; NullSpace; OuterProductMatrix; Permanent; Pivot; QRDecomposition; RandomMatrix; RandomVector; Rank; ReducedRowEchelonForm; Row; RowDimension; RowOperation; RowSpace; ScalarMatrix; ScalarMultiply; ScalarVector; SchurForm; SingularValues; SmithForm; SubMatrix; SubVector; SumBasis; SylvesterMatrix; ToeplitzMatrix; Trace; Transpose; TridiagonalForm; UnitVector; VandermondeMatrix; VectorAdd; VectorAngle; VectorNorm; VectorScalarMultiply; ZeroMatrix; ZeroVector; Zip.

Более подробно с каждой их них можно познакомиться, используя справочную систему HELP.

Возможности создания собственных подпрограмм

Пакет LinearAlgebra предусматривает два уровня обращения к своим процедурам — интерактивный и программный.

Интерактивный уровень

Интерактивный уровень — это дружеский интерфейс к программному уровню. На этом уровне разрешено опускать ненужные параметры, размещать их в любом порядке и не заботится об инициализации объектов. Большинство команд этого уровня предусматривают выполнение трех шагов:

1. Определение типов параметров, переданных пользователем.

2. Задание начальных значений параметрам, опущенным пользователем.

3. Вызов соответствующие подпрограммы программного уровня.

В своих собственных процедурах можно миновать выполнения этих действий, если подготовить все необходимые параметры самостоятельно, и обратиться непосредственно к программным модулям.

Программный уровень

Программный уровень дает возможность наиболее быстро выполнить нужные вычисления. На программном уровне:

- должны быть сформированы и переданы все параметры, которые передаются в строго фиксированном порядке, как того требует синтаксис подпрограммы;
- вызов на программном уровне воспринимается и интерактивным уровнем, чем можно пользоваться при отладке;
- практически все переданные параметры считаются правильно сформированными и не проверяются.

Синтаксические формы вызова всех процедур на программном уровне можно найти в разделе LinearAlgebra Programming Layer Submodule Calling Sequences.

Интерактивному вызову всегда соответствует программный вызов, однако есть два исключения — это вызовы программ, у которых операции зависят от типа передаваемых данных, и команды **Map(..)** и **Map2(..)**. Для первого случая программный вызов сначала обрабатывает переданные параметры, определяя их типы и проверяя данные, а только затем вызывается нужная подпрограмма. То есть фактически эти подпрограммы — это совокупность подпрограмм, работающих с определенными типами данных. Во втором случае даже такой подход реализовать не удается из-за разнообразия возможных входных данных, и поэтому программный вызов совпадает с интерактивным.

Эффективность вычислений

Работа с большими матрицами и двумя типами данных, а именно, с машинными числами с плавающей точкой с ограниченным числом разрядов (машинные вещественные числа) и вещественными числами с плавающей точкой с неограниченным числом разрядов, реализованных программно (программные вещественные числа), осуществляется с помощью подпрограмм предоставленных группой Numerical Algorithms Group (NAG) разработчикам пакета (Waterloo

Maple Inc.). Дополнительно для этих целей использовались подпрограммы из коллекций CLAPACK и ATLAS.

Выбор конкретной подпрограммы определяется следующим:

- типами данных входных объектов;
- опциями **shape**, **storage** и **order,** использованными при определении входных данных;
- значением переменной окружения **UseHardwareFloats**;
- наличием соответствующей программы в библиотеке NAG.

Типы данных

Обращение к подпрограммам библиотеки NAG происходит, только если входные объекты (Matrices, Vectors, or scalars) относятся к типу **numeric**. Для проверки типа коэффициентов сначала анализируется опция **datatype**. Если эта опция говорит, что пакет имеет дело с числовыми данными, то дальнейших проверок не происходит. Если опция **datatype** отсутствует, анализируются сами данные.

Если среди входных данных обнаружено хотя бы одно с плавающей точкой, создаются копии всех остальных данных с другими типами, и они переводятся в числа с плавающей точкой с числом цифр, в соответствии со значениями параметра **Digits** (для случая программных вещественных чисел).

Учет опций Shape, Storage и Order

Подпрограммы пакета NAG предполагают, что все данные расположены в памяти в соответствии с требованиями Фортрана. По умолчанию в Maple это так, но опция **order** может иметь и другое значение, и тогда исходные данные надо скопировать и преобразовать.

Две опции (**shape** и **storage)** матричных объектов позволяют выбрать наиболее эффективную реализацию для этого случая. Обычно задания формы матрицы **shape** достаточно, так как пакет сам подберет нужное расположение матрицы в памяти.

Если подпрограммы NAG все-таки не могут работать с конкретными опциями, заданными пользователем, то происходит копирование данных и их переупорядочивание.

Различные виды чисел с плавающей точкой

Переменная окружения **UseHardwareFloats** контролирует вычисления с числами с плавающей точкой. С ее помощью можно управлять выбором варианта библиотеки NAG для работы с машинными числами (**UseHardwareFloats = true**) или варианта для работы с числами произвольной длины (**UseHardwareFloats = false**). Последний вариант библиотеки был специально построен для работы с программными числами с плавающей точкой.

Если матрица имеет опцию **datatype = float**, проверяется переменная **UseHardwareFloats**, и выясняется, какие вычисления предусмотрены — с машинными числами (**UseHardwareFloats = true**) или программными числами (**UseHardwareFloats = false**). Переменная **datatype = float[8]**, однозначно сообщает, что речь идет о машинных числах, а **datatype = sfloat** — о программных числах**.**

Преимущественное значение для различного задания типов данных имеет значение переменной **UseHardwareFloats**. Например, если **A** и **B** — матрицы с коэффициентами **datatype = sfloat**, **Digits = 100** и **UseHardwareFloats = true**, то матрица **A+B** будет вычислена версией подпрограмм NAG для машинных чисел.

Дополнительные сведенья о форме матриц и способах распределения памяти

В общем случае команды пакета LinearAlgebra используют подпрограммы NAG, когда данные заданы в виде **datatype=sfloat** или **datatype=float[8]**, и опции **shape** (none), **storage** (rectangular). Некоторые команды пакета LinearAlgebra используют подпрограммы библиотеки и при других формах матрицы (например, **triangular[upper]**) и совместимых способах распределения памяти (например,

shape=triangular[upper] и **storage=triangular[upper]** или **storage=rectangular)**.

Во всех остальных случаях происходит копирование входных данных и приведение их к нужной форме.

Решение систем линейных алгебраических уравнений

Пакет «LinearAlgebra» позволяет не только решать систему линейных алгебраических уравнении, но и контролировать каждый этап решения.

По умолчанию Maple использует машинную арифметику чисел с плавающей точкой, реализованную аппаратно, но может использовать вещественные числа, реализованные «программно».

Непосредственное решение с помощью LU-разложения

Решаемое уравнение M.x=V относительно x предполагает задание матрицы системы и вектора правой части:

> **M:=<<0|3|1>,<7|-13|-2>,<1|2|4>>;**

V:=<5.84,-8.46,13.11>;

Для того чтобы решить систему, достаточно воспользоваться командой LinearSolve. Команда позволяет задать систему либо в виде матрицы и вектора, либо расширенной матрицей. Явное указание метода в данном случае не обязательно, так этот выбор осуществляется по умолчанию

> **x := LinearSolve(M, V, method='LU');**

x := LinearSolve(<M|V>, method='LU');

Пошаговое решение системы с помощью LU-разложения

Зададим систему M.x=V:

> **M:=<<0|3|1>,<7|-13|-2>,<1|2|4>>;**

V:=<5.84,-8.46,13.11>;

и построим LU-разложение

P,L,U:=LUDecomposition(M);

где P — матрица перестановок, L — нижняя треугольная матрица с единичной диагональю, U — верхняя треугольная матрица.

Таким образом, мы представляем исходную матрицу в виде трех сомножителей P.L.U=M. Воспользуемся тем, что P — ортогональная матрица, и ее обратная матрица равна ее транспонированной.

> **V2:=Transpose(P).V;**

Таким образом, мы получаем эквивалентную систему L.U.x = Transpose(P).V или L.U.x=V2. Так как L — нижняя треугольная, используем процедуру прямой подстановкой ForwardSubstitute для того чтобы найти вектор V3, такой что L.V3 = V2:

> **V3:=ForwardSubstitute(L,V2);**

Теперь осталась найти вектор x, такой что U.x=V3. Используя обратную подстановку:

>x:=BackwardSubstitute(U,V3);

находим решение.

Аналогично могут быть применены и другие разложения матриц для решения систем.

Как Maple определяет, с какой точностью вычислять

Прежде всего, Maple пытается определить, заданы ли исходные данные в виде вещественных чисел или в виде символьных выражений. Это можно определить, узнав значение параметра datatype матрицы Matrix или проанализировав тип коэффициентов. Для больших матриц последняя операция может занять достаточно много времени, поэтому рекомендуется задавать datatype явно предварительно.

Если входные данные — числа numerics (целые, дроби, числа с плавающей точкой) и переменная **UseHardwareFloats** имеет значение **deduced** (по умолчанию), то используется режим **машинных чисел**.

Если входные данные — числа numerics (целые, дроби, числа с плавающей точкой) и переменная **UseHardwareFloats** имеет значение **false**, то используются программные числа с плавающей точкой —

десятичные числа с плавающей точкой, хранящиеся в виде пары целых чисел (мантисса, показатель).

В остальных случаях применяются символьные вычисления.

Подводя итоги, можно коротко сформулировать преимущества той или иной арифметики (см. табл. П2.16).

Таблица П2.16

Характеристики различных типов счета

Машинные числа	
Преимущества	***Недостатки***
- очень быстро: используется предварительно откомпилированный код (NAG) с аппаратно реализованной арифметикой	- точность вычислений ограничена
Программно реализованные вещественные числа	
Преимущества	***Недостатки***
- достаточно быстро: откомпилированный код (NAG) используется совместно с программно реализованной арифметикой - точностью вычислений можно управлять, используя параметр **Digits**	- не так быстро, как использование аппаратно-реализованной арифметики
Символьные вычисления	
Преимущества	***Недостатки***
- исходные данные и ответ в символьном виде - строится точное решение	- относительно медленно: используется режим интерпретации

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Морозов А. Д.* Инвариантные множества динамических систем в Windows / А. Д.Морозов [и др.]. — М.: Эдиториал УРСС, 1998. — 240 с.

2. *Лойцянский Л. Г.* Курс теоретической механики: в 2 т. / Л. Г. Лойцянский, А. И. Лурье. — Москва, Государственное издательство технико-теоретической литературы, 1954. — 379 с., 1955. — 595 с.

3. *Пановко Я. Г.* Введение в теорию механических колебаний / Я. Г. Пановко. — М.: Наука, 1971. — 240 с.

4. *Ольсон Г.* Динамические аналогии / Г. Ольсон. — М.: Государственное издательство иностранной литературы, 1947. — 224 с.

5. *Higham D. J.* MATLAB guide / D. J. Higham, N. J. Higham. — SIAM, 2000. — 283 с.

6. *Потемкин В.Г.* MATLAB 5 для студентов / В. Г. Потемкин. — М.: ДИАЛОГ–МИФИ, 1998. — 314 с.

7. *Дьяконов В. П.* Математическая система MAPLE V R3/R4/R5 / В. П. Дьяконов. — М.: «СОЛОН», 1998. — 399 с.

8. *Колесов Ю. Б.* Моделирование с систем. Динамические и гибридные системы / Ю. Б. Колесов, Ю. Б. Сениченков. — СПб.: БХВ, 2006. — 224 с.

9. *Колесов Ю. Б.* Моделирование систем. Объектно-ориентированный подход / Ю. Б. Колесов, Ю. Б. Сениченков. — СПб.: БХВ, 2006. — 192 с.