Федеральное агентство по образованию

Нижегородский государственный университет им. Н.И. Лобачевского

Приоритетный национальный проект “Образование”
Инновационная образовательная программа Нижегородского университета:
Образовательно-научный центр
«Информационно-телекоммуникационные системы:
физические основы и математическое обеспечение»

А.В. Островский, А.С. Ефимов,
О.А. Морёнов, А.Н. Половинкин

Математические модели в естествознании и технике

Лабораторный практикум в пакете AnyLogic

Часть 1

Нижний Новгород
Издательство Нижегородского госуниверситета
2007

УДК 5:1 (076)
ББК Бв6я73-4
М 34

Рецензенты:
доктор физ.-мат. наук, профессор М.М. Коган;
кандидат физ.-мат. наук, доцент В.М. Сандалов

М 34 **Математические модели в естествознании и технике. Лабораторный практикум в пакете AnyLogic / А.В. Островский, А.С. Ефимов, О.А. Морёнов, А.Н. Половинкин.** Н. Новгород: Изд-во Нижегородского госуниверситета им. Н.И. Лобачевского, 2007. 123 с.
ISBN 978-5-85746-952-1

Учебное пособие включает лабораторный практикум по математическому моделированию в естествознании и технике с реализацией в программном пакете AnyLogic™, а также справочное руководство по работе с программой. В описание каждой из лабораторных работ включены теоретические основы, схема реализации и практические задания.

Для студентов, обучающихся по специальности и направлению «Прикладная математика и информатика», аспирантов, преподавателей, а также для всех интересующихся математическим моделированием сложных систем.

Издано в рамках Инновационной образовательной программы ННГУ: Образовательно-научный центр «Информационно-телекоммуникационные системы: физические основы и математическое обеспечение»

ББК Бв6я73-4

ISBN 978-5-85746-952-1

© Островский А.В., Ефимов А.С.,
Морёнов О.А., Половинкин А.Н., 2007

© Нижегородский государственный
университет им. Н.И. Лобачевского, 2007

Содержание

Введение 7

1. Основы моделирования динамики систем в программной среде AnyLogic 11

1.1. Общие сведения о возможностях среды AnyLogic ... 11

1.2. Режим разработки модели 13
1.2.1. Панели инструментов 13
1.2.2. Создание новой модели 14
1.2.3. Добавление нового класса активного объекта 16
1.2.4. Добавление нового параметра и изменение настроек параметров 17
1.2.5. Добавление новой переменной и изменение свойств переменных 18
1.2.6. Математические функции 21
1.2.7. Алгоритмические функции 23
1.2.8. Добавление и редактирование кода инициализации и дополнительного кода класса 24
1.2.9. Стейтчарты 25
1.2.10. Создание анимаций 29
1.2.11. Вывод осциллограмм в анимации 34
1.2.12. Агентное моделирование 36
1.2.13. Удаление классов, функций, анимаций и стейтчартов из проекта 36
1.2.14. Работа с файлами и буфером обмена 37
1.2.15. Отмена и возврат последнего действия 37
1.2.16. Подготовка к запуску модели: задание параметров эксперимента 38

1.2.17. Более подробная информация по работе с AnyLogic 38

1.3. Некоторые специальные приемы разработки моделей в AnyLogic 39
1.3.1. Создание диаграммы Ламерея средствами AnyLogic 39
1.3.2. Работа с динамическими объектами в AnyLogic ... 41
1.3.3. Создание простейшей агентной модели в AnyLogic (на примере брюсселятора) 43

1.4. Режим выполнения модели 52
1.4.1. Переключение между режимами разработки и выполнения модели 52
1.4.2. Изучение динамики модели с помощью различных окон 53
1.4.3. Работа с диаграммами 54

2. Простые математические модели сложных динамических систем 57

2.1. Модель динамики уровня зеркала водохранилища с гидростанцией 57
2.1.1. Модель 57
2.1.2. Реализация в AnyLogic 60
2.1.3. Задания для эксперимента 60

2.2. Энергетическая модель сердца 61
2.2.1. Модель 61
2.2.2. Реализация в AnyLogic 64
2.2.3. Задания для эксперимента 65

2.3. Модель засоления (растворимого загрязнения) водоема с заливом 65
2.3.1. Модель 65
2.3.2. Реализация в AnyLogic 71
2.3.3. Задания для эксперимента..... 72

3. Классические эффекты в динамике линейных осцилляторов 73

3.1. Динамика линейного осциллятора..... 73
3.1.1. Модель 73
3.1.2. Реализация в AnyLogic 74
3.1.3. Задания для эксперимента..... 76

3.2. Динамический демпфер 77
3.2.1. Модель 77
3.2.2. Реализация в AnyLogic 79
3.2.3. Задания для эксперимента..... 80

3.3. Связанные осцилляторы 81
3.3.1. Модель 81
3.3.2. Реализация в AnyLogic 84
3.3.3. Задания для эксперимента..... 86

4. Эффекты в нелинейных динамических системах 87

4.1. Динамика электрической схемы с неоновой лампой 87
4.1.1. Модель 87
4.1.2. Реализация в AnyLogic 89
4.1.3. Задания для эксперимента..... 91

4.2. Часы Галилея – Гюйгенса 91
4.2.1. Модель 91
4.2.2. Реализация в AnyLogic 95
4.2.3. Задания для эксперимента..... 96

4.3. Ламповый генератор электрических колебаний..... 96
4.3.1. Модель 96
4.3.2. Реализация в AnyLogic 107
4.3.3. Задания для эксперимента..... 109

4.4. Динамика вертикального полета мячика с отбором энергии в верхней точке 109
4.4.1. Модель 109
4.4.2. Реализация в AnyLogic 115
4.4.3. Задания для эксперимента..... 116

5. Имитационное (агентное) моделирование динамических систем 117

5.1. Имитационная модель «хищник – жертва» с учетом насыщения 117
5.1.1. Модель 117
5.1.2. Реализация в AnyLogic 119
5.1.3. Задания для эксперимента..... 120

5.2. Модель химической кинетики – брюсселятор 120
5.2.1. Модель 120
5.2.2. Реализация в AnyLogic 121
5.2.3. Задания для эксперимента..... 122

Список литературы 123

Введение

В настоящее время возросшие потребности человечества в решении важных и сложных задач из различных областей знаний (физики, химии, биологии, экономики и т.п.) требуют развития научного подхода к решению этих задач. Одним из видов такого подхода является математическое моделирование, при котором реальный объект с помощью некоторого преобразования переводится в операционную математическую систему, затем в этой системе производится исследование модели, и результаты с помощью обратного преобразования интерпретируются с точки зрения реального поведения объекта. Основная трудность здесь заключается в правильном построении математической модели, которая бы адекватно описывала сложный процесс. При этом нужно уметь выделять наиболее существенные аспекты, позволяющие достаточно полно описать динамику реальной системы.

В связи с этим на факультете ВМК Нижегородского государственного университета с 1994 г. читается новый, уникальный курс «Математические модели естествознания» (с 2006 г. – «Концепции современного естествознания (математические модели в естествознании и экологии)»), разработанный профессором ННГУ, академиком РАЕН Ю.И. Неймарком. Курс дает представление о математическом моделировании как науке и искусстве. Здесь строятся и изучаются модели различных систем (физических, химических, биологических, экономических), характеризующиеся различным уровнем сложности и детализации. В процессе изучения моделей дается понятие об основных формах движений системы (свободные и вынужденные колебания, нормальные колебания и биения, автоколебания, хаос) и излагаются основные математические методы исследования систем (методы теории колебаний, теории марковских систем и т.п.). По данному курсу выпущена книга [1], рекомендованная Научно-методическим советом по прикладной математике и информатике УМО университетов РФ в качестве учебника для студентов специальности «Прикладная математика и информатика».

Однако обычное лекционно-книжное изложение такого нестандартного курса, как «Математические модели естествознания», представляется неполным. Желательно, чтобы студенты имели возможность проводить компьютерные эксперименты и наблюдать динамику систем как в виде фазовых траекторий и осциллограмм, так и в форме имитации движений реальных систем (т.е. физику процесса), а также оперативно изменять параметры и начальные условия для наиболее полного исследования возможных вариантов поведения систем и наблюдения бифуркаций.

Кроме того, хотелось бы иметь возможность строить агентные модели сложных систем, изначально не описываемых дифференциальными уравнениями, и исследовать их динамику. Затем, в свою очередь, саму имитационную модель можно было бы представить для студентов как некий объект, для которого предлагается построить обобщенную модель в терминах дифференциальных уравнений.

В связи с этим в настоящем учебном пособии предложен лабораторный практикум по курсу «Концепции современного естествознания (математические модели в естествознании и экологии)», призванный дополнить и расширить рамки, в которых студенты получают представление о математическом моделировании при лекционном изложении данного курса. Компьютерное моделирование позволяет проводить эксперименты с более сложными системами, нежели класс систем, допускающих аналитическое исследование, а также осуществлять математический подход к описанию динамики систем с различных позиций (детерминированные модели, стохастические модели, имитационные модели и т.д.). Для моделирования систем предлагается использовать программный пакет AnyLogic™ компании XJ Technologies (Санкт-Петербург, Россия).

Учебное пособие написано коллективом сотрудников кафедры теории управления и динамики машин и лаборатории «Информационные технологии» (ITLab) факультета вычислительной математики и кибернетики (ВМК) Нижегородского государственного университета (ННГУ) им. Н.И. Лобачевского.

Данное учебное пособие состоит из пяти разделов. В первом из них дается сравнительный анализ возможностей пакета AnyLogic и других известных программных комплексов для построения и исследования моделей динамических систем, а также излагаются основные принципы работы в AnyLogic. Разделы со второго по четвертый посвящены моделированию динамических систем в форме обыкновенных дифференциальных уравнений.

Второй раздел посвящен построению и исследованию простейших качественных моделей динамики достаточно сложных систем (водохранилище с гидростанцией, сердце, процесс засоления водоема с заливом). Из этих моделей, несмотря на их чрезвычайную простоту, можно извлечь большой объем полезной информации (в том числе выработать некоторые рекомендации по управлению параметрами и начальными условиями для достижения оптимального режима работы системы).

Тематика третьего раздела – исследование систем, состоящих из одного или нескольких линейных осцилляторов. Объектом компьютерного исследования здесь является вначале одиночный линейный осциллятор с построением частотных характеристик (АЧХ, ФЧХ, АФЧХ), затем – более сложные системы связанных осцилляторов (два и три связанных осциллятора, динамический демпфер). В конце раздела предлагается смоделировать одномерный волновой процесс в упругой среде путем аппроксимации распределенной среды дискретной системой из конечного числа упруго связанных осцилляторов.

Четвертый раздел посвящен исследованию некоторых физических систем, описываемых нелинейными дифференциальными уравнениями, в которых обнаруживаются эффекты, характерные только для нелинейных систем (существование нескольких устойчивых состояний равновесия, автоколебания, хаотические движения).

В пятом разделе дается понятие о принципиально новом типе моделирования – имитационном, или агентном, и предлагается провести исследование некоторых моделей такого типа (модели химической кинетики, экологические модели типа «хищник – жертва»). Студенты должны подобрать для описания этих систем

адекватные дифференциальные уравнения, динамика которых соответствовала бы динамике рассматриваемой имитационной модели.

Во всех разделах *курсивом* выделены важные моменты изложения материала, на которые нужно обратить особое внимание, а также названия команд (элементов управления) AnyLogic (меню, панели инструментов, кнопки и т.д.). Новые термины, появляющиеся впервые в процессе изложения, выделены **полужирным шрифтом**. Звездочкой (*) отмечены задания повышенной сложности.

Авторский коллектив выражает искреннюю признательность доценту кафедры теории управления и динамики машин факультета ВМК ННГУ С.Ю. Городецкому за неоценимую помощь в организации работы.

1. Основы моделирования динамики систем в программной среде AnyLogic

1.1. Общие сведения о возможностях среды AnyLogic

Для построения и исследования компьютерных моделей различных процессов существует большое количество программных пакетов (TraX, MathCad, MATLAB и т.д.). Эти пакеты позволяют исследовать системы разностных и дифференциальных уравнений достаточно большой размерности, производить различные вычисления с фазовыми переменными и параметрами, строить фазовые портреты, осциллограммы и графики функций. Однако при математическом моделировании реальных систем (физических, химических, биологических, технологических) хотелось бы иметь *наглядное представление* о динамике реальных объектов, т.е. наблюдать на экране некоторую **анимацию**. Во многих существующих программных продуктах, ориентированных на чисто математические расчеты, средства наглядного изображения физического смысла результатов исследования либо *отсутствуют* (TraX), либо *не очень удобны* для создания анимаций (MATLAB), либо *уровень наглядности* анимаций *не очень высок* (MathCad). Кроме того, в этих пакетах отсутствует возможность построения **имитационных** (в том числе **агентных**) моделей, т.е. моделей, в которых параметры и механизмы взаимодействия элементов системы вводятся *напрямую* (без дифференциальных или разностных уравнений, но с указанием некоторых правил взаимодействия).

Все вышеперечисленные возможности, отсутствующие во многих известных компьютерных системах, а также стандартные средства исследования динамических систем реализованы в программной среде AnyLogic™ компании XJ Technologies (Санкт-Петербург, Россия). Данная среда работает в Windows и позволяет:

1. Строить модели процессов в виде динамических систем (в форме обыкновенных дифференциальных уравнений).

2. Строить модели в дискретно-непрерывном времени (имитационные модели, задаваемые диаграммами состояний – стейтчартами). Примерами таких моделей являются модель процесса выливания воды из сосуда через сифон (сифонную трубку) [1] и модель функционирования электрической схемы с неоновой лампой [2]. Дифференциальные уравнения, описывающие процессы в этих системах, зависят от внутреннего состояния сифона или лампы («включено – выключено»).

3. Строить агентные модели, в которых задаются правила взаимодействия агентов (основных элементов функционирования системы). Эти правила определяют поведение системы. Потребность в таких моделях возникает при исследовании динамики сложных биологических и химических систем (взаимодействие популяций, химические реакции), систем массового обслуживания (дорожное движение, работа аэропорта, телефонной станции, многопроцессорной компьютерной системы) и т.д.

4. Выводить на экран фазовые портреты динамических систем и осциллограммы различных переменных и выражений.

5. Создавать анимации для наглядной интерпретации физического смысла моделей. Анимации позволяют не только создавать «мультфильмы», имитирующие работу реальной системы, но и выводить на экран графики различных функций.

6. Писать внутренние (действующие в процессе работы конкретной модели) программные коды на языке Java, что позволяет организовывать структурные вычисления (с использованием программируемых функций), совершенствовать вывод результатов (анимаций, графиков), создавать динамические объекты для промежуточных действий и т.д.

AnyLogic является **объектно-ориентированной** средой для создания моделей. Это означает, что для построения моделей могут быть использованы (а для создания анимаций – *должны* быть использованы) **объектные переменные** (**экземпляры классов**), обладающие набором **полей**, **свойств**, **методов** и **событий**. Но это не просто объектно-ориентированная, а **визуальная** среда: всю работу по созданию объектов AnyLogic берет на себя, а пользователь может изменять их свойства и писать дополнитель-

ные программные коды. Как и во всякой визуальной программной среде, в AnyLogic имеется два режима работы: **режим разработки** и **режим выполнения** модели.

1.2. Режим разработки модели

1.2.1. Панели инструментов

В AnyLogic имеется несколько панелей инструментов:

Стандартные:

Модель **:**

Компиляция:

Вставка:

Рисование:

Структура и стейтчарт:

Анимация:

3D анимация[1]:

Чтобы показать или скрыть панель инструментов, нужно:

1. Выбрать команду меню *Сервис* | *Настройка*….

2. В диалоговом окне «*Настройка*» выбрать вкладку «*Панели инструментов*».

---

[1] В AnyLogic слово «*3D анимация*» написано через пробел, хотя в русском языке обычно пишется через дефис («*3D-анимация»*).

3. Включить или выключить флажок напротив названия нужной панели.

4. Закрыть диалоговое окно «*Настройка*».

1.2.2. Создание новой модели

Модель (**проект**) в AnyLogic – это файл с расширением .ALP (AnyLogic Project – проект AnyLogic). Для создания нового файла проекта нужно:

1. На панели инструментов «*Стандартные*» нажать кнопку (*Создать*) или выбрать команду меню *Файл* | *Создать*..., или нажать `Ctrl+N`.

2. В текстовом поле «*Расположение*» ввести путь к папке, в которую нужно поместить файл, или нажать кнопку «*Выбрать*» для выбора нужной папки.

3. В текстовом поле «*Имя проекта*» ввести имя файла проекта (без расширения .ALP).

4. Нажать кнопку *OK*.

После этого вы увидите в центральной части окна AnyLogic большое белое пустое поле с точками (сетка для привязки объектов) и вкладкой «Main» вверху (рис. 1.1). Это поле **структурной диаграммы** модели, на которой можно размещать **переменные** (п. 1.2.5), а также **стейтчарты** (п. 1.2.9).

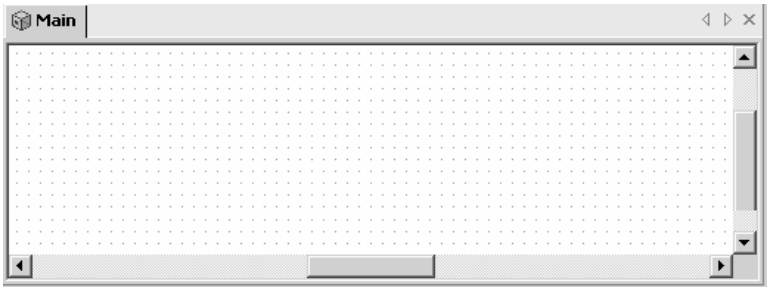

Рис. 1.1

Кроме структурной диаграммы, внутри окна AnyLogic появятся следующие окна:

1. «*Проект*» (рис. 1.2) – окно слева от структурной диаграммы. В этом окне в виде дерева отображается структура проекта (классы, анимации, коды, стейтчарты).

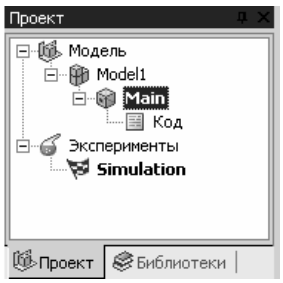

Рис. 1.2

2. «*Свойства*» (рис. 1.3) – окно справа от структурной диаграммы. В этом окне можно устанавливать свойства классов, переменных, объектов для анимации и т.д.

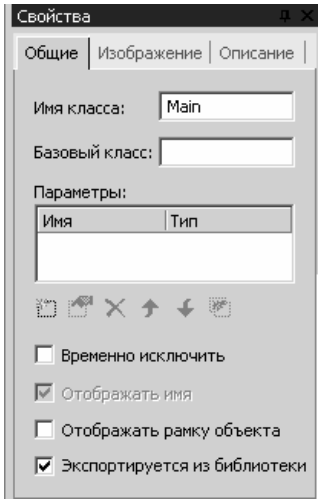

Рис. 1.3

1.2.3. Добавление нового класса активного объекта

В среде AnyLogic реализован визуальный объектно-ориентированный подход, позволяющий организовать удобную структуру модели в виде **классов** (**классы активного объекта**). При создании новой модели автоматически создается главный, или корневой, класс (**корневой объект модели**) с именем `Main`, соответствующий всей модели. Кроме класса `Main`, модель может содержать и другие классы, определяемые пользователем.

Чтобы добавить новый класс, нужно:

1). На панели инструментов «*Вставка*» нажать кнопку (*Новый класс активного объекта*) или выбрать команду меню *Вставка* | *Новый класс активного объекта*….

2). Ввести имя нового класса (из английских букв, цифр и символов подчеркивания, начинающееся не с цифры).

3). Нажать кнопку *OK*.

После этого в окне «*Проект*» появится новый значок для нового класса с соответствующим поддеревом, которое можно раскрывать и сворачивать, щелкая мышью по значкам ⊞ и ⊟ слева от значка . Например, если новому классу было дано имя `MyClass`, то окно «*Проект*» будет выглядеть как на рис. 1.4.

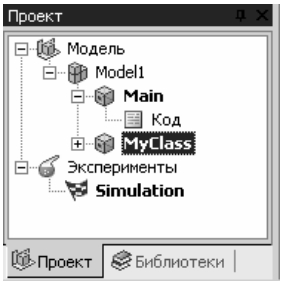

Рис. 1.4

Кроме того, после добавления нового класса (или при двойном щелчке по значку этого класса в окне «*Проект*») вместо

структурной диаграммы всей модели (т.е. главного класса `Main`) появится структурная диаграмма выбранного класса, выделенная рамкой (рис. 1.5).

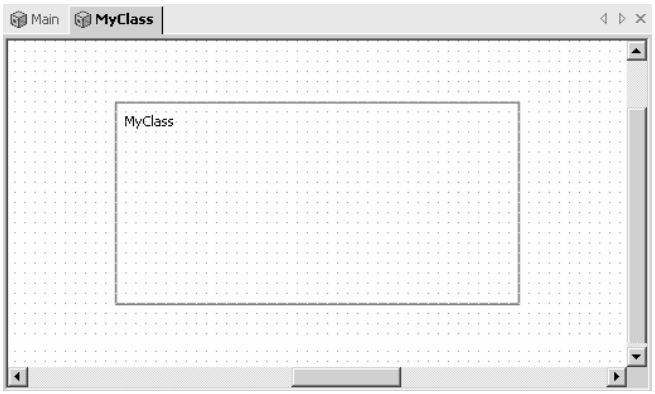

Рис. 1.5

1.2.4. Добавление нового параметра и изменение настроек параметров

Модель может содержать **параметры**. Они могут входить, например, в дифференциальные уравнения (в виде постоянных коэффициентов) или в правила взаимодействия агентов в агентных моделях. При выполнении модели значения параметров могут изменяться с помощью связанных с ними элементов управления в анимации (флажков, ползунков и т.д.) или с помощью дополнительного программного кода. При создании новых классов нужно указать их поля, которые могут определяться в виде параметров, а также в виде переменных (о переменных см. ниже п. 1.2.5).

Чтобы добавить новый параметр в модель, нужно:

1. В окне «*Проект*» щелкнуть по значку того класса, в котором нужно создать параметр или поле (о классах см. п. 1.2.3).

2. В нижней части окна «*Свойства*» нажать кнопку (*Новый параметр*) или нажать клавишу `Insert (Ins)`.

3. В диалоговом окне «*Параметр*» ввести имя параметра (из английских букв, цифр и символов подчеркивания, начинающееся не с цифры), выбрать тип параметра (из выпадающего списка) и ввести начальное значение параметра (в текстовом поле «*По умолчанию*»).

4. В диалоговом окне «*Параметр*» с помощью радиокнопок выбрать тип доступа к параметру (простой, динамический, глобальный, разделитель). По умолчанию выбрана радиокнопка «*Простой*».

5. В диалоговом окне «*Параметр*» по необходимости заполнить поля «*Предопределенные символы*», «*Скрывать если*» и «*Описание*».

6. Нажать кнопку *OK*.

Для изменения настроек конкретного параметра (имя, начальное значение, тип и т.д.) нужно:

1. В окне «*Свойства*» два раза щелкнуть мышью на нужном параметре.

2. Выполнить действия, указанные в пунктах 3 – 6 для добавления нового параметра.

1.2.5. Добавление новой переменной и изменение свойств переменных

Кроме параметров, модель может содержать **переменные**, которые могут изменяться динамически в процессе выполнения модели (например, фазовые переменные или функции от фазовых переменных), а также использоваться для построения графиков функций в режиме выполнения модели. Внутри классов переменные могут использоваться для задания полей класса.

Для добавления новой переменной нужно:

1. В окне «*Проект*» (в левой части окна AnyLogic) два раза щелкнуть мышью по значку того класса, внутри которого нужно

создать переменную (для всей модели – класс с именем `Main`, которое можно поменять в окне «*Свойства*», см. п. 1.2.3).

2. На панели инструментов «*Структура и стейтчарт*» нажать кнопку (*Переменная*) или выбрать команду меню *Рисование* | *Структура* | *Переменная*.

3. Щелкнуть мышью внутри структурной диаграммы того класса, внутри которого нужно создать переменную. Появится новый значок переменной с указанием ее имени (по умолчанию переменные получают имена `var`, `var1`, `var2` и т.д., но эти имена можно изменять в окне «*Свойства*»).

Значки переменных можно передвигать по структурной диаграмме с помощью мыши (при нажатой левой кнопке).

Для изменения свойств конкретной переменной нужно:

1. Выделить эту переменную (щелкнуть по ней на структурной диаграмме соответствующего класса). Справа от структурной диаграммы появится окно «*Свойства*» для переменной. Все последующие действия относятся к окну «*Свойства*».

2. В текстовом поле «*Имя*» ввести (по желанию) новое имя переменной.

3. С помощью радиокнопок выбрать тип переменной: скаляр (тип скаляра выбирается из выпадающего списка), матрица или массив (с указанием размерностей).

4. Из выпадающего списка «*Уравнение* | *Вид*» выбрать вид уравнения, которому будет подчиняться переменная в режиме выполнения. Виды уравнений могут быть следующими:

- «*Интеграл или накопитель*» – для динамической (фазовой) переменной, изменяющейся во времени согласно обыкновенному дифференциальному уравнению первого порядка, разрешенному относительно производной. Правые части дифференциальных уравнений вводятся по общим правилам языка программирования Java с использованием имен переменных и параметров, арифметических операций, круглых скобок и функций. Имена библиотечных функций AnyLogic (математические функции языка Java) можно вставлять с помощью мастера функций (кнопка справа от выпадающего списка «*Вид*») или просто

вводить с клавиатуры. Например, математическая модель маятника с вязким трением и внешним гармоническим воздействием $\ddot{x} + 2\delta\dot{x} + \omega^2 x = F_0 \cos \nu t$ может быть введена в AnyLogic так, как показано на рис. 1.6 (для каждой из двух фазовых переменных `x` и `vx` уравнение вводится в своем окне свойств).

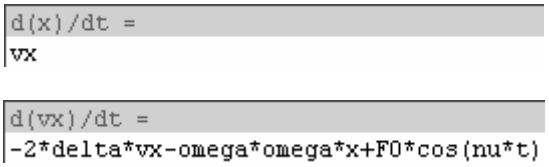

Рис. 1.6

- «*Формула*» – для переменной, значение которой в каждый момент времени вычисляется по определенной формуле. Формула вводится в текстовом поле под выпадающим списком «*Уравнение* | *Вид*» и может включать имена переменных и параметров, арифметические операции, круглые скобки и функции. Имена библиотечных функций AnyLogic (математические функции языка Java) можно вставлять с помощью мастера функций (кнопка f_x справа от выпадающего списка «*Вид*») или просто вводить с клавиатуры. Пусть, например, имеются переменная `x` и переменная `y`, значение которой в каждый момент времени должно быть равно квадрату `x`. Тогда формула для `y` вводится так, как показано на рис. 1.7.

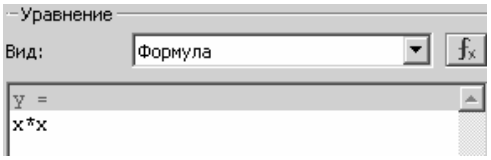

Рис. 1.7

Допускается ввод формул в виде трехместных (тернарных) операций языка Java. В этом случае в условии, заканчивающемся

знаком вопроса, перед именем переменной нужно ставить знак # (диез, номер, «решетка») (рис. 1.8).

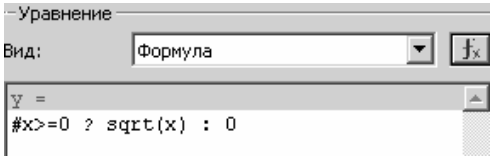

Рис. 1.8

- «*Нет уравнения*» – если переменная не подчиняется никакому уравнению. Эту опцию можно выбрать, например, если значение переменной будет изменяться с помощью дополнительного программного кода.

5. В текстовом поле «*Начальное значение*» ввести начальное значение переменной в виде числа (если выбран вид уравнения «*Интеграл или накопитель*» или «*Нет уравнения*»). Текстовое поле «*Начальное значение*» находится ниже выпадающего списка «*Вид*» (если выбран вид уравнения «*Интеграл или накопитель*», то под текстовым полем для ввода уравнения).

6. Переключиться из окна «*Свойства*» в любое другое окно в пределах окна AnyLogic.

Если какая-либо переменная задается формулой или дифференциальным уравнением, в которое входит другая переменная, то после переключения в структурную диаграмму на ней автоматически появится направленная стрелка, показывающая **связь** между этими переменными. Переменные (в зависимости от вида уравнений и их связей с другими переменными) отображаются на структурной диаграмме различными значками , , .

1.2.6. Математические функции

Математические функции используются в случае, когда одно и то же выражение встречается в модели в нескольких местах (в программном коде, в окнах свойств). Эти функции должны

обязательно возвращать некоторое значение типа `real` (`double`), `integer` (`int`), `Matrix` или `HyperArray`. Тело такой функции должно состоять из одного выражения. Имена математических функций могут состоять из английских букв, цифр и символов подчеркивания и не должны начинаться с цифры. Математические функции в AnyLogic должны быть **методами** внутри некоторого класса (например, класса `Main`).

Для добавления новой математической функции нужно:

1. На панели инструментов «*Вставка*» нажать кнопку (*Новая математическая функция*) или выбрать команду меню *Вставка* | *Новая математическая функция*….

2. В диалоговом окне «*Новая математическая функция*» ввести имя новой математической функции и выбрать из выпадающего списка класс активного объекта, которому должна принадлежать эта функция как метод класса.

3. Нажать кнопку *OK*.

Для редактирования математической функции нужно:

1. Щелкнуть мышью в окне «*Проект*» по значку нужной математической функции. Справа от структурной диаграммы появится окно «*Свойства*» для математической функции. Все последующие действия относятся к окну «*Свойства*».

2. В текстовом поле «*Имя*» ввести имя функции.

3. Из выпадающего списка «*Тип функции*» выбрать тип функции (тип возвращаемого значения).

4. По необходимости ввести список формальных аргументов функции (имена и типы) в группе полей «*Аргументы*».

5. В текстовом поле «*Выражение*» ввести выражение (формулу) для функции. В это выражение можно вставлять библиотечные функции языка Java, в том числе с помощью мастера (кнопка «*Мастер*» расположена ниже текстового поля «*Выражение*»). *Тернарные* операции языка Java *в математических функциях AnyLogic не допускаются*.

6. С помощью флажка «*Статическая функция*» определить, должна ли функция быть статической.

7. Переключиться из окна «*Свойства*» в любое другое окно в пределах окна AnyLogic.

1.2.7. Алгоритмические функции

Если в нескольких местах программы требуется использовать один и тот же *алгоритм действий* (например, для построения графиков функций или для программирования вычислений, которые нельзя описать одной формулой), то этот алгоритм можно задать в виде **алгоритмической функции**. Алгоритмические функции могут возвращать значения типа `real` (`double`), `integer` (`int`), `boolean`, `Matrix`, `HyperArray`, `String`, `Object` или не возвращать значения (тип `void`). Алгоритмические функции в AnyLogic должны быть **методами** внутри некоторого класса (например, класса `Main`).

Для добавления новой алгоритмической функции нужно:

1. На панели инструментов «*Вставка*» нажать кнопку (*Новая алгоритмическая функция*) или выбрать команду меню *Вставка* | *Новая алгоритмическая функция*….
2. В диалоговом окне «*Новая алгоритмическая функция*» ввести имя новой алгоритмической функции и выбрать из выпадающего списка класс активного объекта, которому должна принадлежать эта функция как метод класса.
3. Нажать кнопку *OK*.

Для редактирования алгоритмической функции нужно:

1. Щелкнуть мышью в окне «*Проект*» по значку нужной алгоритмической функции. Справа от структурной диаграммы появится окно «*Свойства*» для алгоритмической функции. Все последующие действия относятся к окну «*Свойства*».
2. В текстовом поле «*Имя*» ввести имя функции.
3. Из выпадающего списка «*Тип функции*» выбрать тип функции (тип возвращаемого значения).
4. По необходимости ввести список формальных аргументов функции (имена и типы) в группе полей «*Аргументы*».

5. В многострочном текстовом поле «*Тело функции*» ввести тело функции в виде программного кода на языке Java. В тело функции можно вставлять библиотечные функции языка Java, в том числе с помощью мастера (кнопка «*Мастер*» расположена ниже текстового поля «*Тело функции*»). Допускается использование тернарных операций языка Java (в отличие от задания переменных в виде формул, в алгоритмических функциях не нужно ставить знак диеза перед именем переменной, входящей в условие со знаком вопроса; сравни п. 1.2.5). Пример приведен на рис. 1.9.

6. С помощью флажка «*Статическая функция*» определить, должна ли функция быть статической.

7. Переключиться из окна «*Свойства*» в любое другое окно в пределах окна AnyLogic.

1.2.8. Добавление и редактирование кода инициализации и дополнительного кода класса

По желанию можно добавить в состав класса специальные программные коды на языке Java: код инициализации и дополнительный код класса.

Код инициализации запускается в режиме выполнения модели сразу после того, как будет создан очередной объект (экземпляр) данного класса. В коде инициализации можно писать, например, операторы присваивания начальных значений полям класса.

В **дополнительном коде класса** можно объявлять новые поля класса и присваивать им начальные значения, описывать новые методы для класса. Таким образом, алгоритмы, работающие внутри класса, можно задать либо в виде математических или алгоритмических функций (пп. 1.2.6, 1.2.7), либо в дополнительном коде класса.

Чтобы добавить или отредактировать код инициализации и дополнительный код класса, нужно:

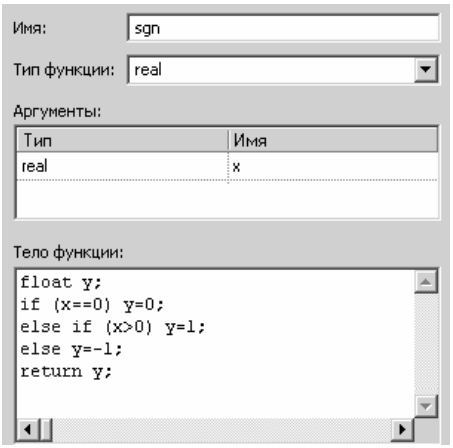

Рис. 1.9

1. В окне «*Проект*» раскрыть дерево нужного класса.

2. Два раза щелкнуть мышью по значку Код , расположенному внутри дерева класса под значком . После этого в центральной части окна AnyLogic вместо структурной диаграммы появится окно для набора программных кодов и уравнений. Все последующие действия относятся к этому окну.

3. В разделе (многострочном текстовом поле) под заголовком «*Код инициализации*» ввести или изменить код инициализации (если нужно).

4. В разделе (многострочном текстовом поле) под заголовком «*Дополнительный код класса*» ввести или изменить дополнительный код класса (если нужно).

1.2.9. Стейтчарты

Во многих реальных случаях работа системы может изменяться при наступлении некоторого события (например, при ударе по летящему мячу, подключении регулятора к сложной системе, по истечении некоторого промежутка времени и т.д.). В этом случае говорят, что система **переходит из одного состояния в**

другое и функционирует в **дискретно-непрерывном времени**. Подход, основанный на понятии **состояния** (не путать с состоянием динамической системы!), позволяет моделировать сложные системы, которые могут функционировать по-разному (в силу различных динамических уравнений), находясь в различных состояниях. Моделирование с использованием состояний – это одна из разновидностей **имитационного моделирования**.

Поведение системы, которая может находиться в различных состояниях и переходить из одного состояния в другое, может быть задано в среде AnyLogic графически с помощью **стейтчартов** (от английского statechart – диаграмма состояний). Стейтчарт создается на структурной диаграмме класса и представляет собой ориентированный граф, вершинами которого являются **состояния**, а ребрами – стрелки, задающие **переходы** между состояниями (частный случай перехода – **петля-переход**, т.е. переход из какого-либо состояния в это же состояние). Каждое состояние и каждая стрелка должны быть полностью описаны.

Для создания стейтчарта нужно:

1. На панели инструментов «*Структура и стейтчарт*» нажать кнопку (*Стейтчарт*) или выбрать команду меню *Рисование* | *Структура* | *Стейтчарт*.
2. Щелкнуть мышью внутри структурной диаграммы нужного класса в том месте, куда следует поместить стейтчарт.

Стейтчарты можно передвигать по структурной диаграмме с помощью мыши (при нажатой левой кнопке).

Чтобы добавлять новые состояния и описывать переходы, необходимо открыть **диаграмму стейтчарта**. Для открытия диаграммы стейтчарта нужно два раза щелкнуть мышью по значку стейтчарта , расположенному в окне «*Проект*» или на структурной диаграмме класса. После этого в центральной части окна AnyLogic вместо структурной диаграммы класса появится диаграмма стейтчарта, которая изначально состоит из **начального состояния** (по умолчанию состояния получают имена `state`,

`state1`, `state2` и т.д.) и **указателя начального состояния** (рис. 1.10).

Рис. 1.10

Чтобы добавить новое состояние на диаграмму стейтчарта, нужно:

1. На панели инструментов «*Структура и стейтчарт*» нажать кнопку (*Состояние*) или выбрать команду меню *Рисование* | *Стейтчарт* | *Состояние.*

2. Щелкнуть мышью на диаграмме стейтчарта в том месте, куда нужно поместить состояние.

Чтобы изменить имя состояния, нужно:

1. Щелкнуть мышью по нужному состоянию на диаграмме стейтчарта. Состояние будет выделено.

2. В окне «*Свойства*» в текстовом поле «*Имя*» ввести новое имя состояния (по общим правилам образования имен переменных).

Состояния можно передвигать по диаграмме стейтчарта с помощью мыши (при нажатой левой кнопке).

Чтобы добавить новый переход между состояниями, нужно:

1. На панели инструментов «*Структура и стейтчарт*» нажать кнопку (*Переход*) или выбрать команду меню *Рисование* | *Стейтчарт* | *Переход.*

2. Подвести курсор мыши к рамке того состояния, из которого нужно сделать переход (курсор мыши должен принять форму кружка с черточками внутри). Затем щелкнуть левой кнопкой мыши.

3. Двигать курсор мыши по направлению к тому состоянию, в которое нужно сделать переход (без удержания кнопок мыши). Стрелка перехода автоматически проследует за курсором мыши. Если нужно добавить изгиб стрелки, то необходимо щелкнуть

левой кнопкой мыши в том месте, где нужно создать изгиб, и затем двигать курсор мыши дальше.

4. Подвести курсор мыши к рамке того состояния, в которое нужно сделать переход (курсор мыши должен принять форму кружка с черточками). Затем щелкнуть левой кнопкой мыши. В результате будет добавлен переход в виде стрелки.

Для **описания перехода** (т.е. для задания действий, которые должны выполняться при наступлении перехода) нужно:

1. Щелкнуть мышью по нужному переходу на диаграмме стейтчарта. Переход будет выделен. Все дальнейшие действия относятся к окну «*Свойства*».

2. В текстовом поле «*Имя*» по необходимости ввести новое имя перехода (по умолчанию переходы получают имена `transition`, `transition1`, `transition2` и т.д.).

3. С помощью выпадающего списка «*Происходит*» указать, в каком случае срабатывает переход (возможны пять вариантов: «*Немедленно*», «*По таймауту*[2]», «*По сигналу*», «*Когда условие становится истинным*», «*По событию*»). Ниже этого списка расположена группа текстовых полей (разделов): верхнее поле (его имя меняется в зависимости от выбранного варианта в списке «*Происходит*»), а также необязательные для заполнения поле «*Гард*» и многострочное поле «*Действие*».

4. В верхнем поле (см. п. 3) в зависимости от выбранного случая срабатывания перехода задать либо таймаут (в единицах модельного времени), либо сигнал (сигналом может служить объект любого класса Java), либо условие или событие для срабатывания (в виде логического условия на языке Java). Если в выпадающем списке «*Происходит*» выбрана опция «*Немедленно*», то верхнее поле не заполняется.

---

[2] Таймаут (англ. timeout) – задержка выполнения программы в течение определенного промежутка времени. В программе AnyLogic и *Руководстве пользователя* (см. п. 1.2.17) слово «таймаут» написано слитно, хотя в русском языке обычно пишется через дефис («тайм-аут»). В настоящем пособии сохранена орфография AnyLogic.

5. По необходимости в поле «*Гард*» ввести **гард**. Гард (от английского guard – охрана, в данном случае – охраняющее условие) – это логическое условие, разрешающее или запрещающее переход. Если это условие истинно, то переход разрешается, а если ложно, то переход запрещается. Гарды пишутся по обычным правилам записи условий на языке Java.

6. По необходимости в поле «*Действие*» ввести программный код на языке Java, задающий действия, которые должны выполниться при срабатывании перехода.

Пример приведен на рис. 1.11 (здесь `flag1` – имя переменной логического типа, т.е. типа `boolean`).

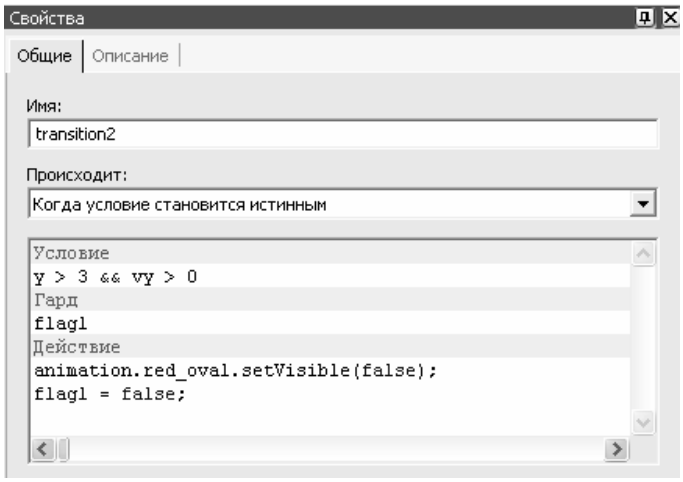

Рис. 1.11

1.2.10. Создание анимаций

Среда AnyLogic позволяет наглядно интерпретировать физический смысл динамики моделей. Для этого имеется возможность построения **анимаций**. Анимация представляет собой окно, в котором могут размещаться следующие элементы:

1. **Геометрические фигуры**, из которых строится собственно анимированное изображение. Движение, форму и изменение цвета геометрических фигур в анимации можно задавать, привязывая их к некоторым выражениям, зависящим от значений фазовых переменных и параметров модели.

2. **Диаграммы**, позволяющие выводить данные о динамике системы (осциллограммы, дуговые и столбцовые диаграммы) внутри анимации.

3. **Элементы управления**, позволяющие изменять параметры и переменные, входящие в модель. Таким образом, анимации в AnyLogic являются **интерактивными (управляемыми)**.

Чтобы добавить новую двумерную (2D) анимацию, нужно:

1. На панели инструментов «*Вставка*» нажать кнопку (*Новая анимация*) или выбрать команду меню *Вставка* | *Новая анимация*… .

2. В диалоговом окне «*Новая анимация*» ввести имя новой анимации и выбрать из выпадающего списка класс активного объекта, которому должна принадлежать эта анимация (каждый класс может содержать только одну анимацию).

3. Нажать кнопку *OK*.

Чтобы добавить новую трехмерную (3D) анимацию, нужно:

1). На панели инструментов «*Вставка*» нажать кнопку (*Новая 3D анимация*) или выбрать команду меню *Вставка* | *Новая 3D анимация*… .

2). В диалоговом окне «*Новая 3D анимация*» ввести имя новой 3D-анимации и выбрать из выпадающего списка класс активного объекта, которому должна принадлежать эта анимация (каждый класс может содержать только одну 3D-анимацию).

3). Нажать кнопку *OK*.

Все геометрические фигуры, диаграммы и элементы управления являются объектами Java и помещаются в анимацию визуально с помощью мыши и панели инструментов «*Анимация*». В этом пункте будут рассмотрены только двумерные (2D) анимации (именно они используются в данном лабораторном практи-

куме). Трехмерные (3D) анимации создаются аналогично с помощью панели инструментов «*3D анимация*».

Панель инструментов «*Анимация*» показана ниже.

В нижеследующей таблице перечислены кнопки этой панели (слева направо) с указанием визуальных объектов, вставляемых в анимацию с помощью этих кнопок, а также имен типов классов этих объектов в Java и имен, которые получают эти объекты в AnyLogic по умолчанию.

Кнопка	Объект (элемент анимации)	Имя класса в Java	Имена переменных в AnyLogic по умолчанию
	Прямоугольник	`ShapeRect`	`rectangle`, `rectangle1`, …
	Скругленный прямоугольник	`ShapeRoundRect`	`roundRect`, `roundRect1`, …
	Овал	`ShapeOval`	`oval`, `oval1`, …
	Ломаная линия	`ShapePoly`	`polyLine`, `polyLine1`, …
	Линия (прямолинейный отрезок)	`ShapeLine`	`line`, `line1`, …
	Текст (статический текст)	`ShapeText`	`text`, `text1`, …
	Группа фигур	`Group`	`pivot`, `pivot1`, …
	Изображение (для загрузки картинок)	`ShapeImage`	`image`, `image1`, …
	Вложенная анимация	Производный от `ShapeBase`	`encapsulatedAnimation`, `encapsulatedAnimation1`, …
	Дуговой индикатор	`ArcIndicator`	`arcIndicator`, `arcIndicator1`, …

	Столбцовый индикатор	`BarIndicator`	`barIndicator`, `barIndicator1`, …
	Индикатор-диаграмма	`ChartIndicator`	`chartIndicator`, `chartIndicator1`, …
	Кнопка	`ShapeButton`	`button`, `button1`, …
	Флажок	`ShapeCheckbox`	`checkBox`, `checkBox1`, …
	Переключатели (радио-кнопки)	`ShapeGroupbox`	`radioButtons`, `radioButtons1`, …
	Бегунок (слайдер)	`ShapeSlider`	`slider`, `slider1`, …
	Поле ввода	`ShapeTextInput`	`editBox`, `editBox1`, …
	Перемещаемый прямоугольник	`ShapeHandle`	`handle`, `handle1`, …

Сама анимация также является визуальным объектом класса `AnimationPanel` и по умолчанию имеет имя `animation` (как поле внутри некоторого класса, например класса `Main`).

Более подробная информация о классах, их свойствах и методах содержится в *Справочнике классов AnyLogic*, который вызывается командой меню *Справка* | *Справочник классов*.

Чтобы добавить новый визуальный объект в анимацию, нужно:

1. Открыть нужную анимацию с помощью двойного щелчка по значку анимации в окне «*Проект*». В центральном окне вместо структурной диаграммы появится **анимационная диаграмма**, выделенная пунктирной рамкой.

2. На панели инструментов «*Анимация*» нажать кнопку с изображением нужного объекта, или выбрать соответствующую команду вертикального меню, вложенного в пункт меню *Рисование* | *Анимация*.

3. С помощью мыши нарисовать объект в нужном месте внутри анимационной диаграммы. При этом:

а) Для рисования линии (`ShapeLine`) нужно вначале щелкнуть мышью, затем вести линию и по окончании рисования опять щелкнуть мышью.

б) Для рисования ломаной линии (`ShapePoly`) нужно вначале щелкнуть мышью, затем по одному разу щелкать мышью по окончании рисования каждого отрезка ломаной линии. Для окончания рисования ломаной линии нужно два раза щелкнуть мышью.

в) Все остальные визуальные объекты можно рисовать либо одним щелчком мыши, либо ведением мыши по анимационной диаграмме при нажатой левой кнопке.

Чтобы удалить визуальный объект из анимации, нужно:

1. Открыть анимацию так же, как при добавлении нового объекта в анимацию.

2. Выделить нужный объект с помощью щелчка мыши.

3. Нажать клавишу `Delete (Del)`.

Чтобы изменить свойства конкретного визуального объекта, входящего в анимацию, нужно:

1. Открыть анимацию так же, как при добавлении нового объекта в анимацию.

2. Выделить нужный объект с помощью щелчка мыши.

3. В диалоговом окне «*Свойства*» изменить свойства объекта с помощью элементов управления (текстовых полей, выпадающих списков и флажков; для каждого класса визуальных объектов – свой набор элементов управления). Элементы управления в окне «*Свойства*» появляются при щелчке на вкладке «*Общие*», а также на вкладке, имя которой совпадает с именем класса выбранного объекта («*Линия*», «*Ломаная*» и т.д.). При этом имеются следующие особенности:

а) Некоторые свойства (например, координаты концов линии) можно сделать как **статическими** (равными некоторым константам), так и **динамическими** (определяемыми по формулам, в которые входят переменные и параметры). Если одновременно за-

даны и статическое значение, и динамическая формула, то свойство будет изменяться по динамической формуле (т.е. динамическое задание «сильнее» статического).

б) Для ломаной линии можно задавать координаты ее **точек** (т.е. концов отрезков ломаной) общей формулой, в которую входит номер точки ломаной линии (номер точки обозначается словом `index`). Точки ломаной линии нумеруются, начиная с нуля, в том порядке, в котором они были нарисованы.

в) Элементы управления позволяют генерировать **события** (щелчок по кнопке, перемещение бегунка и др.), которые можно обрабатывать в окне «*Свойства*» с помощью программного кода в разделе (многострочном текстовом поле) «*Реакция на событие*». Кроме того, бегунки можно связать с параметрами и переменными (щелчок в окне «*Свойства*» по вкладке «*Бегунок*», далее выбирать переменную из выпадающего списка «*Переменная*»), что позволяет изменять значения переменных и параметров непосредственно в режиме выполнения модели с помощью бегунка.

На рис. 1.12 показан пример заполнения окна *«Свойства»* для прямоугольника, а на рис. 1.13 – для ломаной линии.

1.2.11. Вывод осциллограмм в анимации

Для сравнения математических (численных) результатов исследования модели с ее физической интерпретацией рекомендуется одновременно с движением геометрических фигур выводить на экран осциллограммы (графики изменения фазовых переменных во времени). Осциллограммы можно выводить на экран либо внутри анимации, либо с помощью отдельных окон для диаграмм. О выводе осциллограмм в отдельных окнах будет рассказано в п. 1.4.3.

Для вывода осциллограмм внутри анимации используются объекты «Индикатор-диаграмма» (`ChartIndicator`). Добавление индикатора-диаграммы в анимацию производится по обычным правилам добавления визуальных объектов (п. 1.2.10).

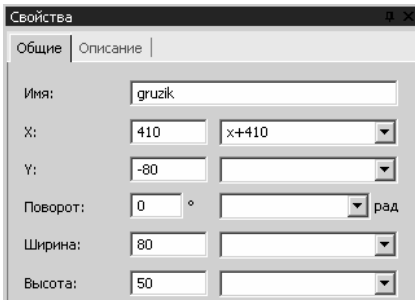

Рис. 1.12

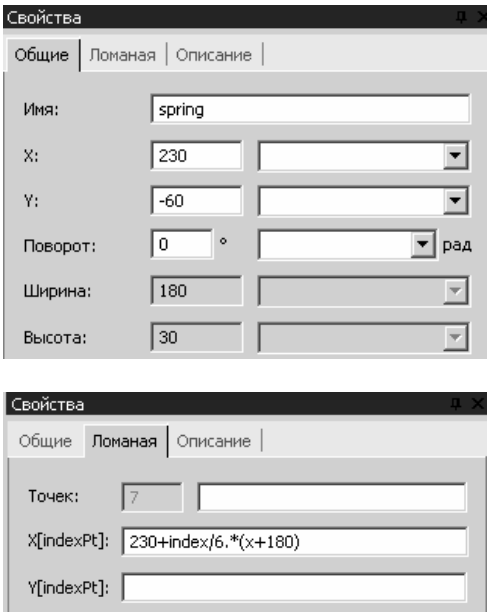

Рис. 1.13

Параметры индикатора-диаграммы (выводимая переменная, размер окна, цвет линий) устанавливаются в окне «*Свойства*»

(для появления этого окна нужно выделить индикатор-диаграмму с помощью щелчка мыши). В окне «*Свойства*» нужно выбрать вкладку «*Индикатор-диаграмма*» и изменять параметры с помощью текстовых полей и выпадающих списков. Выводимая переменная выбирается из выпадающего списка «*Отображает*». В группе радиокнопок «*Тип*» должна быть выбрана радиокнопка «*График*». Флажок «*Отображать шкалу*» позволяет включать и выключать отображение шкалы (делений) на графике.

1.2.12. Агентное моделирование

При исследовании динамики различных систем сложной структуры (биологических, химических, экономических, систем массового обслуживания и др.) используется подход, называемый **агентным моделированием.** Используя агентные модели, можно получить общее представление о поведении моделируемой системы в целом, задавая правила поведения ее элементов (**агентов**) и не имея сведений о каких-то общих законах поведения системы в целом. Правила поведения агентов могут задаваться с использованием стейтчартов (п. 1.2.9), а также планирования **событий** (например, с помощью таймеров). Таким образом, агентные модели принципиально отличаются от моделей в виде динамических систем.

Пример построения агентной модели подробно рассмотрен в п. 1.3.3.

1.2.13. Удаление классов, функций, анимаций и стейтчартов из проекта

Если какие-либо элементы проекта (классы, функции, анимации, стейтчарты) становятся ненужными, то их можно удалить или временно исключить из проекта (временное исключение подразумевает возможность возвращения исключенного элемента в проект).

Чтобы временно исключить элемент из проекта, нужно:

1. В окне «*Свойства*» щелкнуть правой кнопкой мыши по значку нужного элемента.

2. В контекстном меню выбрать команду «*Временно исключить*».

Для возвращения исключенного элемента в проект нужно выполнить те же действия, что и при временном исключении.

Для удаления элемента из проекта нужно:

1. В окне «*Свойства*» щелкнуть правой кнопкой мыши по значку нужного элемента.

2. В контекстном меню выбрать команду «*Удалить*».

3. Подтвердить удаление нажатием кнопки «*Да*» в окне сообщения Windows.

1.2.14. Работа с файлами и буфером обмена

Работа с файлами производится так же, как и в любом известном Windows-приложении (Word, Excel и др.), с помощью кнопок панели инструментов «*Стандартные*» или команд меню «*Файл*».

Работа с буфером обмена (Clipboard) производится так же, как и в любом известном Windows-приложении, с помощью кнопок панели инструментов «*Стандартные*» или команд меню «*Правка*». Можно использовать также стандартные комбинации клавиш Windows:

`Ctrl+X` или `Shift+Delete` – вырезать;

`Ctrl+C` или `Ctrl+Insert` – копировать;

`Ctrl+V` или `Shift+Insert` – вставить.

1.2.15. Отмена и возврат последнего действия

Если какое-либо действие (например, ввод или удаление символа либо визуального объекта) произведено ошибочно, то его можно отменить. Для этого нужно нажать комбинацию клавиш `Ctrl+Z` (стандартная комбинация Windows) или `Alt+Backspace` (комбинация AnyLogic) либо выбрать команду меню *Правка* | *Отменить*.

Если действие было ошибочно отменено, то его можно вернуть (повторить). Для этого нужно выбрать команду меню *Правка* | *Вернуть*.

1.2.16. Подготовка к запуску модели: задание параметров эксперимента

Прежде чем запустить модель на выполнение (в терминах AnyLogic – запустить **эксперимент**), нужно задать параметры эксперимента. Для задания параметров эксперимента нужно:

1. В окне «*Проект*» щелкнуть мышью по значку эксперимента (по умолчанию эксперимент имеет имя `Simulation`).

2. В окне «*Свойства*» выбрать вкладку «*Общие*», затем по необходимости задать имя эксперимента, скорость моделирования (виртуальное или реальное время), значения параметров.

3. В окне «*Свойства*» выбрать вкладку «*Дополнительные*», затем по необходимости задать временно́й интервал (текстовое поле «*Условие остановки модели* | *Стоп по времени*»), параметры датчика случайных чисел, численные методы решения уравнений, абсолютную и относительную точность, временну́ю точность, шаг по времени.

1.2.17. Более подробная информация по работе с AnyLogic

Более подробную информацию, касающуюся работы с AnyLogic (типы переменных, углубленные сведения о стейтчартах, экспериментах и т.д.) можно получить в *Руководстве пользователя (User's Manual)*, которое вызывается из AnyLogic командой меню *Справка* | *Руководство пользователя* или *Справка* | *User's Manual*.

1.3. Некоторые специальные приемы разработки моделей в AnyLogic

1.3.1. Создание диаграммы Ламерея средствами AnyLogic

При исследовании динамических систем часто возникает необходимость наряду с фазовым портретом строить точечное отображение, порождаемое фазовыми траекториями (функцию последования). В случае одномерного отображения (отображение кривой в кривую, в частном случае – прямой в прямую) его динамика графически изображается в виде диаграммы Ламерея. Для создания диаграмм Ламерея в AnyLogic может использоваться следующий подход.

На анимационной диаграмме рисуем две оси с помощью линий (`ShapeLine`), добавляем две надписи с названием осей. Добавляем на анимационную диаграмму ломаную линию (`ShapePoly`) с именем `lamLine`, с помощью которой в дальнейшем и будет строиться сам график точечного отображения.

В раздел «*Дополнительный код класса*» добавляем следующий код:

```
private double x0 = <значение>;
private double y0 = <значение>;
/* координаты точки, соответствующей началу
   координат, на анимационной диаграмме */

private double width = <значение>;
private double height = <значение>;
/* ширина и высота диаграммы Ламерея на
   анимационной диаграмме */

private double max_val = <значение>;
/* максимальное абсолютное значение отображаемой на
   диаграмме Ламерея величины */
private double values[]; /* массив для последова-
тельных значений величины, отображаемой на диаграм-
ме Ламерея */
```

```
private int points_num; /* максимальное число точек
ломаной, служащей для отображения диаграммы Ламерея
*/
```

Следующая функция служит для добавления нового значения на график точечного отображения:

```
private void AddNewLamPoint()
{
  if (points_num == 0)
  {
    values[0] = <значение>;
    lamLine.addPoint(x0 + values[0] / max_val *
      width, y0);
    points_num = 1;
  }
  double new_x = 0.;
  double new_y = 0.;
  double curr_val = <значение>; /* текущее значение
    величины, отображаемой на диаграмме Ламерея */
 values[points_num] = curr_val;
  if (curr_val > max_val)
  {
    max_val *= Math.ceil(curr_val / max_val);
    RecalcPoints();
  }
  new_x = x0 + values[points_num - 1] / max_val *
    width;
  new_y = y0 + curr_val / max_val * height;
  lamLine.addPoint(new_x, new_y);
  lamLine.addPoint(x0 + new_y, new_y);
  points_num++;
}
```

В процессе моделирования работы какой-либо физической системы возможна ситуация, когда значение отображаемой физической величины превзойдет `max_val` – в этом случае необходимо произвести масштабирование графика, для чего служит следующая функция.

```
private void RecalcPoints()
```

```
{
  double new_x = 0., new_y = 0.;
  lamLine.setPoints(0);
  lamLine.addPoint(x0 + values[0] / max_val *
    width, y0);
  for (int i = 1; i < points_num; i++)
  {
    new_x = x0 + values[i - 1] / max_val * width;
    new_y = y0 + values[i] / max_val * height;
    lamLine.addPoint(new_x, new_y);
    lamLine.addPoint(x0 + new_y, new_y);
  }
}
```

Функцию добавления новой точки на диаграмму Ламерея следует вызывать следующим образом. В модель добавляем стейтчарт. В состояние, в котором обычно находится модель, добавляем петлю-переход. Выбираем из выпадающего списка *«Происходит»* пункт *«Когда условие становится истинным»* и в разделе *«Условие»* пишем условие перехода, а в разделе *«Действие»* добавляем строку:

```
animation.AddNewLamPoint();
```

1.3.2. Работа с динамическими объектами в AnyLogic

При разработке моделей вам может понадобиться создать некоторое заранее неизвестное число графических объектов и отображать их в анимации. Это может потребоваться, например, при рисовании сложной кривой по точкам (в качестве точек можно использовать объект `ShapeRect` с подобранными соответствующим образом параметрами).

Информацию о создаваемых объектах необходимо где-то хранить, чтобы потом можно было оперировать созданными объектами. Для этого можно было бы использовать массив, но это потребует дополнительных усилий при написании вспомогательного кода для работы с этим массивом. Гораздо быстрее и удобнее воспользоваться готовым классом AnyLogic `SmartArray`

(следует отметить, что данный класс присутствует только начиная с версии 5.2). Поэтому в первую очередь создаем объект класса `SmartArray`. Для этого в раздел «*Дополнительный код класса*» добавляем строку:

```
public static SmartArray arr = new SmartArray();
```

Теперь в этом массиве вы можете хранить информацию обо всех создаваемых объектах. Но рекомендуется в одном массиве хранить объекты одного типа или назначения. Это не строгое требование, а лишь рекомендация, но это позволит вам избежать множества ошибок при написании кода.

Далее в одной из своих функций вы можете динамически создавать объекты, изменять их свойства и запоминать их в созданном ранее массиве. Например:

```
ShapeRect r1 = new ShapeRect(100,200,1,1);
// создание прямоугольника

r1.setColor(Color.blue); // задание цвета
arr.addLast(r1);  // добавление в массив
```

Следует отметить, что вы должны будете сами в коде задать все необходимые визуальные свойства создаваемого объекта (положение на экране, размеры, цвет и т.д.). Информацию о свойствах интересующего вас класса и методах для работы с ними можно посмотреть в *Справочнике классов AnyLogic* (команда меню *Справка* | *Справочник классов*).

И последнее и самое важное, что осталось вам сделать, – это отобразить созданный объект на экране. Для этого надо воспользоваться методом `add` класса `AnimationPanel`:

```
animation.add(r1); // добавление в анимацию
```

При этом отображение нового объекта на экране произойдет автоматически.

И, наконец, может понадобиться удалить все созданные объекты. Для этого, во-первых, необходимо удалить объекты из ани-

мации, используя информацию, сохраненную в массиве. Например:

```
for (int i=0;i<arr.size();i++) // цикл по массиву
   animation.remove((ShapeRect)arr.get(i));
   // удаление из анимации
```

И, во-вторых, необходимо очистить массив. Проще всего это сделать с помощью соответствующей функции:

```
arr.clear();   // очистка массива
```

1.3.3. Создание простейшей агентной модели в AnyLogic (на примере брюсселятора)

С использованием агентных моделей можно получить общее представление о поведении моделируемой системы в целом, задавая правила поведения ее элементов (агентов) и не имея сведений о каких-либо общих законах поведения системы в целом.

Данный подход продемонстрируем на примере одной из моделей химической кинетики – **модели брюсселятора** – с использованием средства моделирования AnyLogic, поддерживающего способы задания поведения объектов как на основе диаграмм состояний (стейтчартов), так и с применением планирования событий.

Химические уравнения модельной реакции, получившей название «брюсселятор» и протекающей в **открытой системе** (исходные реагенты могут подводиться в систему извне, а конечные продукты реакции выводятся из системы), имеют вид [3]:

$$A \rightarrow X,\ B+X \rightarrow Y+D,\ 2X+Y \rightarrow 3X,\ X \rightarrow E.$$

Здесь A и B – исходные реагенты, D и E – конечные продукты реакции, X и Y – промежуточные продукты реакции.

Создание любой агентной модели начинается с определения и создания самих агентов, для которых задаются правила их взаимодействия с другими агентами. В рассматриваемой модели брюсселятора в качестве агентов выступают молекулы.

Для создания агентов в AnyLogic создадим класс активного объекта `Molecule` и затем добавим агенты (объекты этого класса) в модель. Для того чтобы добавить агенты в модель, нужно просто добавить объект класса, задающего структуру агента (`Molecule`), в класс, моделирующий окружение агентов (`Main`). В данной модели необходимо добавить 4 типа агентов (типов молекул или веществ): A, B, X, Y (рис. 1.14).

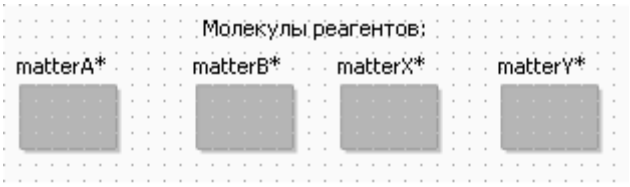

Рис. 1.14

На вкладке «*Количество*» окна «*Свойства*» задайте «*Количество объектов*» A: 800; B: 50; X: 0; Y: 0. Указанное количество объектов класса `Molecule` будет автоматически создано в начале работы модели. Для идентификации типа вещества в класс `Molecule` необходимо добавить параметр `Type` типа `integer` и для указанных четырех типов веществ в таблице «*Параметры*» задать соответствующее значение параметра `Type`: 1, 2, 3 или 4. Также в класс `Main` следует добавить четыре переменные `countA`, `countB`, `countX`, `countY` типа `integer` с нулевыми начальными значениями (эти переменные хранят количества молекул соответствующего типа).

Для упрощения доступа к окружению (объекту класса `Main`) из кода объектов, моделирующих агенты, в класс `Molecule` добавим переменную `model`, указав в качестве ее типа `Main`, а в качестве начального значения `(Main)getOwner()`.

Теперь необходимо определить структуру для представления взаимодействия объектов в данной пространственной модели (два измерения). Для этого будем предполагать, что молекулы взаимодействуют в некоторой квадратной области, которую мы разобьем на виртуальные ячейки. При этом для однозначного оп-

ределения положения молекулы в объеме необходимо задать ее абсолютные координаты. Для этого в класс `Molecule` добавим пару координат `X` и `Y` типа `real`, а также для упрощения моделирования взаимодействия добавим в класс `Molecule` переменную `Cell` типа `integer`, определяющую номер ячейки, в которой находится молекула (нумерация ячеек последовательно по строкам). Кроме того, для моделирования плавного перемещения молекул в объеме в класс `Molecule` также добавим переменные `oldX, oldY` типа `real` и переменную `tmoved` типа `real` с начальным значением -0.02 (данный подход описан в *Agent-Based Modeling Tutorial – Учебном пособии по агентному моделированию*, которое вызывается из AnyLogic командой меню *Справка* | *Agent-Based Modeling Tutorial*).

Параметры для определения количества ячеек, покрывающих область, определим в классе `Main`: `CellWidth` – ширина (высота) одной ячейки, тип – `real`, значение по умолчанию – 20; `Width` – количество ячеек по горизонтали (вертикали), тип – `integer`, значение по умолчанию – 15.

Для того чтобы можно было оперировать с молекулами указанных типов, необходимо каким-либо образом хранить все молекулы. Для этого в классе `Main` определим четыре вектора `MoleculesAInCell, MoleculesBInCell, MoleculesXInCell` и `MoleculesYInCell` (рис. 1.15). Каждый из этих векторов имеет тип `Vector[]`, а также начальное значение, определяемое динамически: `new Vector[Width*Width]`. Таким образом, это векторы, содержащие векторы молекул соответствующего типа в каждой ячейке.

Инициализацию указанных векторов необходимо произвести в коде инициализации класса `Main`:

```
// инициализация вектора для каждой ячейки
for ( int i = 0; i < Width * Width; i++ ) {
  MoleculesAInCell[i] = new Vector();
  MoleculesBInCell[i] = new Vector();
  MoleculesXInCell[i] = new Vector();
  MoleculesYInCell[i] = new Vector();
}
```

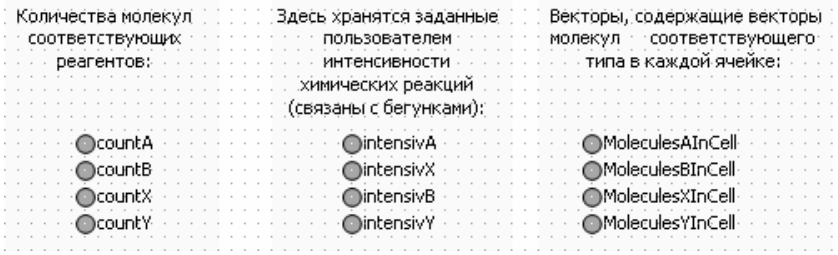

Рис. 1.15

Этап создания агентов определим с помощью стейтчарта (рис. 1.16).

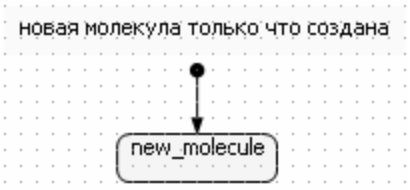

Рис. 1.16

Здесь на переходе в состояние `new_molecule` (молекула только что создана) выполняется следующий код:

```
switch(Type)
{
  case 1: model.MoleculesAInCell[Cell].add(
     Molecule.this ); model.countA++;  break;
  case 2: model.MoleculesBInCell[Cell].add(
     Molecule.this ); model.countB++; break;
  case 3: model.MoleculesXInCell[Cell].add(
     Molecule.this ); model.countX++; break;
  case 4: model.MoleculesYInCell[Cell].add(
     Molecule.this ); model.countY++; break;
}
model.setModified();
```

Таким образом, здесь происходит только добавление вновь созданных молекул в соответствующий вектор и изменение пе-

ременной, хранящей количество молекул. При этом значения переменных вновь созданной молекулы определяются в коде инициализации класса `Molecule` (координаты определяются случайным образом, по ним рассчитывается номер ячейки, в которой оказалась молекула):

```
X = uniform( 0, model.CellWidth * model.Width );
Y = uniform( 0, model.CellWidth * model.Width );
oldX = uniform( 0, model.CellWidth * model.Width );
oldY = uniform( 0, model.CellWidth * model.Width );
int numb_x = ((int)X) / ((int)model.CellWidth);
int numb_y = ((int)Y) / ((int)model.CellWidth);
Cell = numb_y * (model.Width) + numb_x;
```

Теперь определим способы взаимодействия молекул в процессе протекания реакций, определяемых моделью брюсселятора. Определим интенсивности четырех типов реакций в модели как переменные класса `Main` типа `real`: `intensivA, intensivB, intensivX, intensivY`. Значения по умолчанию для них зададим равными соответственно 0.1, 0.2, 0.05, 0.04.

Для каждой реакции (A→X, B+X→Y+D, 2X+Y→3X, X→E) определим соответствующий таймер, генерирующий события: `reaction1, reaction2, reaction3` и `reaction4` (рис. 1.17). Указанные таймеры срабатывают по таймауту; при этом в качестве значения таймаута рассматривается величина, обратная интенсивности соответствующей реакции.

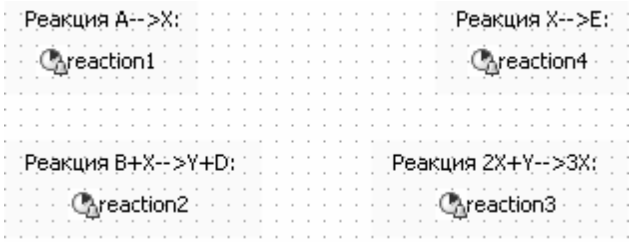

Рис. 1.17

При этом реакции типа 1 и типа 4 более просты в реализации,

поскольку в них взаимодействия молекул не происходит. Рассмотрим в качестве примера действие при срабатывании первого таймера:

```
if ( Type == 1 )
{
  /*  запоминаем значения переменных "исчезающей"
      молекулы типа A */
  double xx = X;
  double yy = Y;
  int celll = Cell;

  /* так как кол-во молекул A не должно измениться,
     то имитируем появление новой молекулы вместо
     "исчезающей" молекулы */
  X = uniform( 0, model.CellWidth * model.Width );
  Y = uniform( 0, model.CellWidth * model.Width );
  int numb_x = ((int)X) / ((int)model.CellWidth);
  int numb_y = ((int)Y) / ((int)model.CellWidth);
  Cell = numb_y * (model.Width) + numb_x;
  oldX = uniform( 0, model.CellWidth *
    model.Width );
  oldY = uniform( 0, model.CellWidth *
    model.Width );
  tmoved = -0.02;

  // создать новую молекулу типа X
  Molecule mol = new Molecule();

  // добавить новую молекулу типа X в модель
  model.setup_matterX(mol,model.matterX.size());

  /* поместить новую молекулу типа X в то же место,
     где была "исчезнувшая" молекула типа A */
  mol.X    = xx;
  mol.Y    = yy;
  mol.Cell = celll;
  mol.Type = 3;

  model.setModified(); }
```

Реакции типа 2 и 3 более сложны, поскольку в них участвует несколько типов молекул. В частности, для реакции 2 код имеет вид:

```
/* проверяется возможность протекания данной
   реакции – наличие необходимого количества
   молекул в данной ячейке */
if ( (Type == 2) &&
  (model.MoleculesXInCell[Cell].size() > 0) )
{
  /* молекула, взаимодействующая с данной
     молекулой, выбирается случайно
     из молекул нужного типа в данной ячейке */
  Molecule other =
   (Molecule)model.MoleculesXInCell[Cell].get(
     uniform_discr(
      0, model.MoleculesXInCell[Cell].size()-1));

  double xx = X;
  double yy = Y;
  int celll = Cell;

  X = uniform( 0, model.CellWidth * model.Width );
  Y = uniform( 0, model.CellWidth * model.Width );
  int numb_x = ((int)X) / ((int)model.CellWidth);
  int numb_y = ((int)Y) / ((int)model.CellWidth);
  Cell = numb_y * (model.Width) + numb_x;
  OldX = uniform( 0, model.CellWidth * model.Width
    );
  oldY = uniform( 0, model.CellWidth * model.Width
    );
  tmoved = -0.02;

  // создать новую молекулу типа Y
  Molecule mol = new Molecule();

  /* добавить новую молекулу типа Y в модель
     (на место "исчезающей" молекулы B) */
  model.setup_matterY(mol,model.matterY.size());
  mol.X    = xx;
```

```
  mol.Y    = yy;
  mol.Cell = celll;
  mol.Type = 4;

  // удалить молекулу типа X
  model.MoleculesXInCell[celll].remove(other);
  model.dispose_matterX( other );
  model.countX--;

  model.setModified();
}
```

Перемещение молекул определяется на основе событий таймера `moveMolecule`, срабатывающего по таймауту с треугольной функцией распределения `uniform (0.01, 0.02)`:

```
oldX = X;
oldY = Y;

tmoved = getTime();
/*  вызвать функцию определения соседней ячейки
    случайным образом */

int cell_numb = model.RandomCellAround(Cell);
X = model.XGlobal(cell_numb,
  uniform(0, Main.CellWidth));
Y = model.YGlobal(cell_numb,
  uniform(0, Main.CellWidth));

// удалить молекулу из рассматриваемой ячейки
switch(Type)
{
  case 1: model.MoleculesAInCell[Cell].remove(
    Molecule.this ); break;
  case 2: model.MoleculesBInCell[Cell].remove(
    Molecule.this ); break;
  case 3: model.MoleculesXInCell[Cell].remove(
    Molecule.this ); break;
  case 4: model.MoleculesYInCell[Cell].remove(
    Molecule.this ); break;
}
```

```
// добавить молекулу в соседнюю ячейку
Cell = cell_numb;
switch(Type)
{
  case 1: model.MoleculesAInCell[Cell].add(
    Molecule.this );  break;
  case 2: model.MoleculesBInCell[Cell].add(
    Molecule.this );  break;
  case 3: model.MoleculesXInCell[Cell].add(
    Molecule.this );  break;
  case 4: model.MoleculesYInCell[Cell].add(
    Molecule.this );  break;
}
model.setModified();
```

Используемая при этом функция определения соседней ячейки `RandomCellAround` взята из модели `PredatorPrey`, поставляющейся вместе с системой AnyLogic.

На анимации для данной модели в виде точек (маленьких прямоугольников) изображены молекулы различных веществ (для каждого вещества – свой цвет) (рис. 1.18). Количество и расположение этих точек-молекул меняются в соответствии с динамикой модели. Принцип подготовки анимации для агентных моделей повторяет подход, подробно описанный в *Agent-Based Modeling Tutorial.*

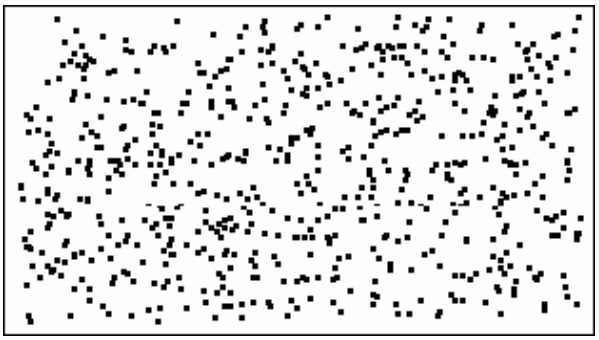

Рис. 1.18

1.4. Режим выполнения модели

1.4.1. Переключение между режимами разработки и выполнения модели

После того как разработка модели окончена, можно **запустить модель на выполнение** (перейти из **режима разработки** в **режим выполнения** модели). В AnyLogic можно осуществить два вида запуска модели:

1. **Обычный запуск** – модель выполняется в течение заданного интервала времени (интервал времени – один из параметров эксперимента, см. п. 1.2.16) либо до принудительного прерывания с помощью кнопок «*Пауза*», «*Перезапустить*» или «*Остановить*» (об этих кнопках см. ниже). Для (обычного) запуска модели нужно на панели инструментов «*Модель*» нажать кнопку (*Запустить*) или выбрать команду меню *Модель* | *Запустить*, или нажать клавишу `F5`. После этого будет затрачено некоторое время на компиляцию модели, и в случае успеха модель будет запущена. В случае ошибки компиляция будет аварийно завершена с выдачей сообщения в поле «*Описание*», расположенном под структурной диаграммой модели.

2. **Запуск на один шаг** (модель выполняется в течение одного шага численного метода). Для такого запуска нужно, находясь уже в **режиме выполнения модели**, на панели инструментов «*Модель*» нажать кнопку (*Выполнить шаг*) или выбрать команду меню *Модель* | *Выполнить шаг*, или нажать клавишу `F10`.

Находясь в **режиме разработки модели**, можно обновить проект и создать модель без запуска. Для этого нужно выполнить те же действия, что и при запуске модели на один шаг. Создание модели без запуска используется, например, если требуется исследовать модель (в том числе построить осциллограммы и фазовый портрет) при конкретных значениях параметров и начальных условий, предварительно установив их в анимации с помощью бегунков.

В **режиме выполнения модели** можно управлять выполнением модели с помощью следующих операций:

1. Пауза (приостановление выполнения модели) – с помощью кнопки (*Пауза*) на панели инструментов «*Модель*» или команды меню *Модель* | *Пауза*, или комбинации клавиш `Ctrl+F10`. Для возобновления выполнения нужно выполнить те же действия, что и для запуска модели.

2. Перезапуск (все элементы модели уничтожаются и заново создаются, модельное время сбрасывается в нуль, шаги не выполняются) – с помощью кнопки (*Перезапустить*) на панели инструментов «*Модель*» или команды меню *Модель* | *Перезапустить*, или комбинации клавиш `Ctrl+Shift+F5`. Перезапуск осуществляется, например, если требуется исследовать модель (в том числе построить осциллограммы и фазовый портрет) при конкретных значениях параметров и начальных условий, предварительно установив их в анимации с помощью бегунков (как и создание модели без запуска).

3. Принудительная остановка (прекращение, прерывание) выполнения модели – с помощью кнопки (*Остановить*) на панели инструментов «*Модель*» или команды меню *Модель* | *Остановить*, или комбинации клавиш `Shift+F5`. Остановка модели означает уничтожение всех элементов модели и переход из режима выполнения в режим разработки модели.

1.4.2. Изучение динамики модели с помощью различных окон

В режиме выполнения модели можно изучать ее динамику как с помощью наглядной анимации, так и с помощью окон, содержащих чисто математическую информацию (окно корневого объекта модели, диаграммы).

Окно корневого объекта модели (`root`) позволяет отслеживать изменение переменных и параметров, являющихся полями корневого класса (объекта). Для отображения этого окна нужно выбрать команду меню *Вид* | *Корневой объект модели*.

Если в режиме выполнения модели нужно видеть на экране **структуру модели** (т.е. структурную диаграмму корневого клас-

са), то нужно выбрать команду меню *Вид* | *Структура*. В этом окне можно войти в структуру других классов с помощью двойного щелчка мышью по значку этого класса.

Для вывода осциллограмм и фазовых портретов (в отдельных окнах) используются окна **диаграмм** (в отличие от **индикаторов-диаграмм**, расположенных внутри анимации, **диаграммы** – это отдельные окна). Добавление новой диаграммы производится с помощью команды меню *Вид* | *Новая диаграмма*. Более подробно о работе с диаграммами будет рассказано в п. 1.4.3.

Между панелями инструментов и окнами, в которых наблюдается динамика или структура модели, расположены вкладки. Их названия соответствуют названиям окон: «Анимация», «Новая диаграмма», «`root`» (окно корневого объекта модели), «`root.vx (root.x)`» (диаграмма для вывода фазового портрета на плоскости переменных (`x`,`vx`)) и т.д. Переключение между окнами в режиме выполнения моделей производится с помощью щелчка на нужной вкладке.

Если какое-либо окно стало ненужным, то его можно убрать с экрана, нажав в этом окне кнопку закрытия.

1.4.3. Работа с диаграммами

Диаграммы в режиме выполнения модели служат для вывода осциллограмм и фазовых портретов. Если нужно вывести на экран не *осциллограмму*, а *фазовый портрет на фазовой плоскости* (а в случае многомерной динамической системы – проекцию фазовых траекторий на какую-либо плоскость), то следует использовать только *окна диаграмм*, так как *индикаторы-диаграммы* (входящие в состав анимации) не позволяют выводить графики, в которых по осям координат не отображается время.

Пусть на экране в режиме выполнения модели имеется диаграмма. Чтобы указать, *какие именно переменные требуется выводить в этой диаграмме*, нужно:

1. Щелкнуть правой кнопкой мыши внутри диаграммы.

2. В контекстном меню выбрать команду «*Содержимое диаграммы…*». Появится диалоговое окно «*Содержимое диаграммы*», в котором имеется три списка: «*Переменные, параметры, наборы данных*», «*Ось Y*», «*Ось X*». Между списками «*Переменные, параметры, наборы данных*» и «*Ось Y*», а также между списками «*Переменные, параметры, наборы данных*» и «*Ось X*» имеется по одной группе из двух кнопок со стрелками вправо и влево. Каждая «стрелка вправо» предназначена для добавления переменной на соответствующую ось, а каждая «стрелка влево» – для отмены вывода переменной по соответствующей оси.

3. Выделяя нужные переменные в списках «*Переменные, параметры, наборы данных*», «*Ось Y*» и «*Ось X*» и нажимая нужные кнопки со стрелками вправо и влево, добавлять или отменять вывод переменных по соответствующим осям. Если по одной оси выводится несколько переменных, то на диаграмме будут отображены несколько графиков разного цвета (цвет каждого графика автоматически помещается в **легенду диаграммы**, расположенную в нижней части диаграммы).

4. С помощью группы радиокнопок «*Отображать*» выбрать, какие именно графики нужно отображать на диаграмме: «*Последний прогон*», «*Все прогоны*», «*Одна точка на прогон*».

5. Нажать кнопку *OK*.

Управление выводом графиков на диаграммах (тип диаграммы, масштабирование осей, цвет линий, вид соединения точек, цвет и стиль маркера изображающей точки) осуществляется следующим образом:

1. Щелкнуть правой кнопкой мыши внутри диаграммы.

2. В контекстном меню выбрать команду «*Параметры диаграммы…*». Появится диалоговое окно «*Параметры диаграммы*». Все последующие действия относятся к этому окну.

3. Выбрать вкладку «*Общие*», далее выбрать тип диаграммы и дополнительные параметры отображения. Чтобы осциллограммы и фазовые портреты выводились в виде кривых, нужно в группе радиокнопок «*Тип диаграммы*» выбрать радиокнопку «*График*» (сравни п. 1.2.11 – параметры индикатора-диаграммы).

4. Выбрать вкладку «*Оси*», далее задать параметры масштабирования, размер окна, цвет сетки и шрифт.

5. Выбрать вкладку «*Наборы данных*», далее задать стиль соединения точек и стиль маркера изображающей точки, цвет линии и цвет маркера.

6. Нажать кнопку *OK*.

2. Простые математические модели сложных динамических систем

2.1. Модель динамики уровня зеркала водохранилища с гидростанцией

2.1.1. Модель

Динамическая модель, описывающая процесс изменения уровня воды в водохранилище с плотиной и гидростанцией (гидротурбина с электрогенератором) (рис. 2.1), имеет вид:

$$S(H)\frac{dH}{dt}=\begin{cases} Q-I(H)-F(H)-\dfrac{W}{k\rho g(H+h)} & (H>0), \\ Q-I(H)-F(H) & (H<0). \end{cases} \qquad (2.1)$$

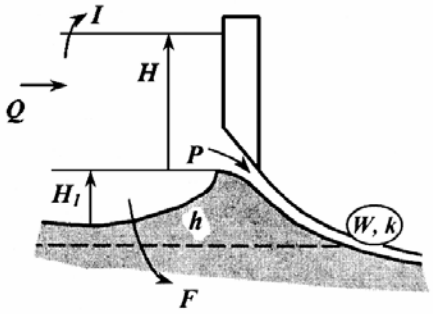

Рис. 2.1

Здесь:

H – уровень воды в водохранилище (отсчитывается от точки входа воды из водохранилища в трубопровод, соединяющий водохранилище с гидротурбиной);

$S(H)$ – площадь поверхности зеркала водохранилища (в простейшем случае – монотонно возрастающая или неубывающая функция от H);

$I(H)$ и $F(H)$ – интенсивности (объемные скорости) испарения и фильтрации соответственно (монотонно возрастающие функции от H);

Q – интенсивность (объемная скорость) притока воды в водохранилище извне, например из рек или с дождями (в простейшем случае – константа, в общем случае – функция времени);

W – мощность электроэнергии, которую требуется вырабатывать;

k – коэффициент полезного действия гидротурбины вместе с электрогенератором;

ρ – плотность воды;

g – ускорение свободного падения;

h – высота точки отсчета величины H относительно гидротурбины (перепад уровня воды перед гидротурбиной равен $H+h$).

Составная правая часть уравнения (2.1) связана с тем, что при опускании уровня воды ниже точки входа в трубопровод ($H<0$) вода не поступает на гидротурбину и слагаемое $\dfrac{W}{k\rho g(H+h)}$ (интенсивность расхода воды на вращение турбины) исчезает.

В уравнение (2.1) со знаком «плюс» входит постоянная (не зависящая от H) величина $F_1(H)\equiv Q$, а со знаком «минус» — функция

$$F_2(H)=\begin{cases} I(H)+F(H)+\dfrac{W}{k\rho g(H+h)} & (H>0), \\ I(H)+F(H) & (H<0). \end{cases}$$

Таким образом, уравнение (2.1) можно переписать в виде:

$$S(H)\frac{dH}{dt}=F_1(H)-F_2(H)\,.$$

Естественно считать, что функции $I(H)$ и $F(H)$ монотонно возрастают с ростом H. Изобразим на одном рисунке графики функций $F_1(H)$ и $F_2(H)$ и проведем при различных фиксиро-

ванных значениях H вертикальные стрелки от графика $F_2(H)$ к графику $F_1(H)$, направление которых показывает, какой знак имеет производная dH/dt при различных значениях H (вверх — положительная, вниз — отрицательная).

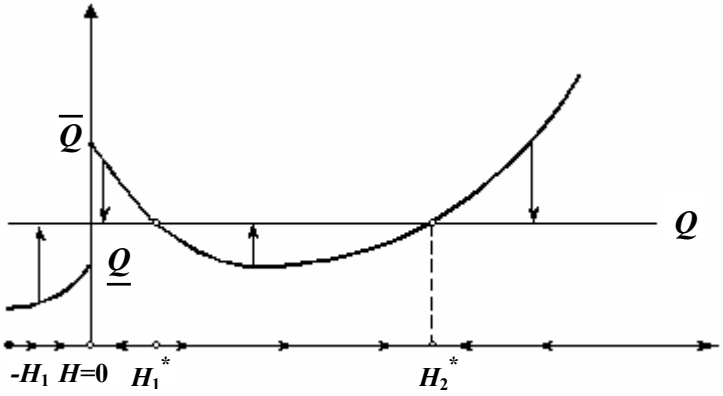

Рис. 2.2

Стрелки на оси H на рис. 2.2 показывают направление изменения H во времени (фазовые траектории уравнения (2.1)). Из рисунка видно, что данная динамическая система имеет два устойчивых состояния равновесия: $H=0$ и $H=H_2^*>0$, между которыми находится неустойчивое состояние равновесия $H=H_1^*$. (Отметим, что точка $H=0$ является, по сути, не состоянием равновесия, а разновидностью «скользящего режима», возникающего из-за конечного разрыва величины dH/dt в силу уравнения (2.1) в точке $H=0$, когда при $H\to -0$ и $H\to +0$ величина dH/dt стремится к ненулевым значениям разных знаков.) При этом в состоянии равновесия $H=0$ гидростанция вырабатывает не полную требуемую мощность W, а только ее часть, равную величине $k\rho gh$ (т.е. $k\rho g(H+h)$ при $H=0$), которая меньше, чем W. В состоянии же равновесия $H=H_2^*$ полностью вырабатывает-

ся требуемая мощность W. Таким образом, равновесие $H = H_2^*$ является желательным режимом работы гидростанции, а $H = 0$ — нежелательным.

Из рис. 2.2 видно, что во избежание прихода траектории в нежелательный режим $H = 0$ нельзя допускать, чтобы величина притока воды Q оказалась меньше некоторого критического уровня $\underline{Q}$, при котором график функции $F_1(H) \equiv Q$ касается графика функции $F_2(H)$. Напротив, если величина Q превзойдет другое пороговое значение $\overline{Q} = F_2(0)$, то состояние равновесия $H = H_2^*$ будет *глобально асимптотически устойчивым*, т.е. при любом начальном уровне воды в водохранилище гидростанция гарантированно войдет в желательный режим работы (область значений $Q > \overline{Q}$ можно назвать поэтому *зоной безопасности*).

2.1.2. Реализация в AnyLogic

Работа реализована в файле `Part1\electrostation.alp` (рис. 2.3). В окне анимации можно наблюдать фазовый портрет на луче $H \geq 0$ на фоне графика правой части уравнения (2.1), а также осциллограмму $H(t)$ и картинку водохранилища с вращающейся турбиной гидростанции (уровень воды в водохранилище и скорость вращения турбины связаны с величиной H). Профиль дна выбран в виде определенной функции, качественно удовлетворяющей свойствам поверхности зеркала реальных водохранилищ. С помощью бегунков пользователь может изменять начальные условия (бегунок под фазовым портретом) и параметры системы.

2.1.3. Задания для эксперимента

1. Изменяя начальные условия, приведите систему (точку на фазовом портрете) в желательный режим работы гидростанции. Какими для этого должны быть начальные условия?

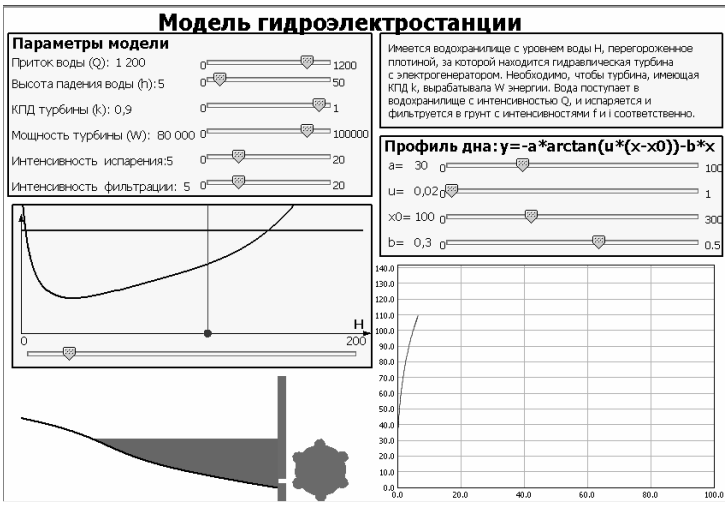

Рис. 2.3

2. Изменяя параметры модели (приток воды, КПД турбины и т.д.), проследите за бифуркациями системы (рождение и исчезновение состояний равновесия). Какие параметры и в какую сторону нужно изменять, чтобы гидростанция гарантированно вошла в зону безопасности?

3. Как влияют параметры профиля дна на динамику системы (в частности, на критические уровни и зону безопасности)?

2.2. Энергетическая модель сердца

2.2.1. Модель

Данная простейшая модель функционирования сердца относится к классу **управляемых динамических систем**, т.е. таких систем, у которых оператор изменения состояния x зависит не только от самого состояния, но и от **управления** – некоторой величины u (в общем случае векторной), которая является функцией времени t. В случае непрерывности времени и дифференци-

руемости оператора управляемая динамическая система задается дифференциальным уравнением $\dot{x}(t) = F(x(t), u(t))$.

В данной модели состояние динамической системы состоит только из одной величины – энергетического запаса сердца Q. Роль управления $u(t)$ (в данном случае – скалярной функции от t) играют сигналы от нервной системы (вегетативной и центральной). Модель имеет вид дифференциального уравнения:

$$\frac{dQ}{dt} = -a - f(u,Q) + g(u,Q), \qquad (2.2)$$

где:

a – интенсивность расхода энергетического запаса на поддержание жизненного состояния неработающего сердца;

$f(u,Q)$ – интенсивность расхода энергетического запаса на совершение работы по перекачке крови;

$g(u,Q)$ – интенсивность пополнения энергетического запаса за счет поступающей в сердце крови.

По физическому (биологическому) смыслу величины u и Q могут меняться в некоторых пределах $0 \le u \le u_{\max}$ и $0 \le Q \le Q_{\max}$ соответственно. Величина $u(t)$ в простейшем случае может быть константой, в общем случае это функция времени.

Очевидно, что при отсутствии управления ($u = 0$), а также при отсутствии энергетического запаса ($Q = 0$) энергия не будет ни расходоваться на перекачку крови, ни пополняться за счет притока крови, т.е. $f(0,Q) = f(u,0) = g(0,Q) = g(u,0) = 0$. Очевидно также, что $f(u,Q)$ — неубывающая функция обоих аргументов.

Так как Q не может быть больше $Q_{\max}$, то при $Q = Q_{\max}$ имеем $dQ/dt \le 0$ и поэтому $-a - f(u,Q_{\max}) + g(u,Q_{\max}) \le 0$.

Естественно предположить, что при $u = u_{\max}$ (т.е. при максимальных требованиях к работе сердца) сердце быстро истощается и будем иметь $dQ/dt < 0$.

С учетом всего вышесказанного получаем, что величина dQ/dt в силу уравнения (2.2) при выполнении хотя бы одного из условий $Q=0$, $u=0$ и $u=u_{\max}$ будет строго отрицательной, а при $Q=Q_{\max}$ — неположительной. Пусть $u(t)\equiv\text{const}$, тогда на плоскости (Q,u) соответствующая фазовая траектория лежит на горизонтальной прямой (рис. 2.4).

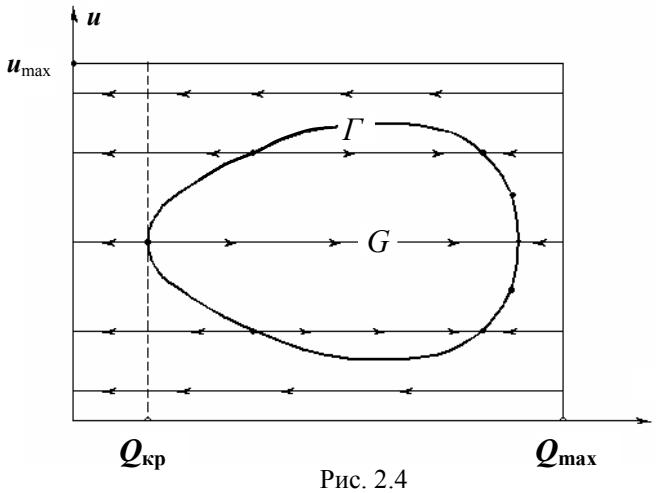

Рис. 2.4

«Область жизни» G (область, в которой величина dQ/dt положительна) ограничена некоторой замкнутой кривой в силу вышеуказанных свойств функций $f(u,Q)$ и $g(u,Q)$. Граница этой области задается уравнением $-a-f(u,Q)+g(u,Q)=0$ и целиком состоит из состояний равновесия (неустойчивых в левой части кривой и устойчивых в правой). При этом существует некое критическое значение энергетического запаса $Q_{\text{кр}}$ такое, что при $Q<Q_{\text{кр}}$ никакое управление не поможет сердцу выжить ($Q\to 0$ с течением времени).

Из рис. 2.4 видно, что для обеспечения жизнедеятельности сердца нужно изменять значения управления u в разумных пре-

делах (в частности, по необходимости уменьшать или, наоборот, стимулировать нагрузку на сердце), а также тренировать сердце для уменьшения критического значения $Q_{кр}$.

2.2.2. Реализация в AnyLogic

Работа реализована в файле `Part1\heart.alp` (рис. 2.5).

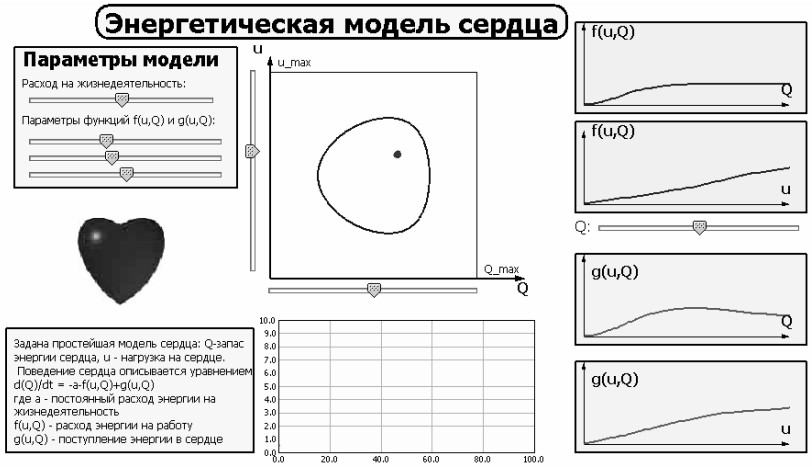

Рис. 2.5

В окне анимации изображена плоскость (Q,u), где с помощью бегунков можно изменять начальный запас энергии сердца Q и значение управления u. Функции $f(u,Q)$ и $g(u,Q)$ в данной реализации зависят от трех параметров (их можно менять с помощью бегунков), причем сумма $f(u,Q)+g(u,Q)$ не зависит явно от этих параметров. Поэтому при изменении параметров функций $f(u,Q)$ и $g(u,Q)$ будут перерисовываться только графики этих функций, но не граница «области жизни» на плоскости (u,Q). Граница «области жизни» перерисовывается при изменении величины a (расхода на поддержание жизнедеятельности неработающего сердца).

В процессе счета (интегрирования) модели по плоскости (u, Q) движется точка, показывающая изменение энергетического запаса сердца. Под бегунками, изменяющими параметры, находится анимированная картинка сердца, которое бьется при $Q > 0$ и перестает биться при $Q = 0$.

2.2.3. Задания для эксперимента

1. Изменяя параметры функций $f(u,Q)$ и $g(u,Q)$, проследите за изменением графиков этих функций. Сделайте вывод о влиянии этих параметров на характер изменения интенсивностей расхода и пополнения энергетического запаса при изменении управления и самого энергетического запаса.

2. Изменяя значения управления u и начального энергетического запаса Q, проследите за динамикой модели. Как не дать сердцу погибнуть?

3. Изменяя величину a, проследите за изменением «области жизни». Существует ли такое значение a, при котором нормальная работа сердца практически невозможна?

2.3. Модель засоления (растворимого загрязнения) водоема с заливом

2.3.1. Модель

Пусть имеется водоем с заливом, причем вода может перетекать только из водоема в залив, но не в обратную сторону. Примером такой водной системы является Каспийское море с заливом Кара-Богаз-Гол, расположенным примерно на 4,5 м ниже уровня Каспия (рис. 2.6). Пусть V и V_1 – объемы воды в море и заливе, Q – интенсивность (объемная скорость) втекания воды в море (реки, дожди, сточные воды), ν – концентрация растворимых примесей (соли и загрязняющих веществ) во втекающем в море потоке воды, I и I_1 – интенсивности (объемные скорости) испарения воды из моря и залива, q — интенсивность (объемная

скорость) перетекания воды из моря в залив. Кроме того, пусть M и M_1 — общие массы растворимого загрязнения в море и заливе, а μ — концентрация загрязнения в море. Считая, что загрязнение в *водоеме* не достигает насыщения, имеем $\mu = M / V$. В *заливе* же насыщение имеет место уже давно, и в нем образовались гигантские донные отложения солей натрия, магния и др.

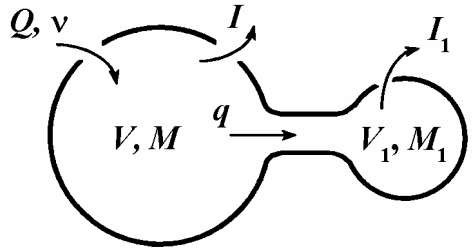

Рис. 2.6

Тогда математическая модель динамики уровней воды и концентраций растворимых примесей в море и заливе имеет вид:

$$\left.\begin{aligned} \frac{dV}{dt} &= Q - I - q, \\ \frac{dV_1}{dt} &= q - I_1, \\ \frac{dM}{dt} &= \nu Q - \mu q, \\ \frac{dM_1}{dt} &= \mu q. \end{aligned}\right\} \qquad (2.3)$$

Интенсивность втекания воды в море Q, вообще говоря, может изменяться и зависит от многих причин (изменение температуры, влажности воздуха и др.), но среднее значение Q достаточно постоянно во времени, поэтому будем считать, что $Q \equiv \text{const}$. Интенсивность испарения воды из моря I является функцией от V, интенсивность испарения воды из залива I_1 — функцией от V_1, а интенсивность перетекания воды из моря в залив q — функцией

от V и V_1. Сделаем достаточно очевидное допущение, что функции $I(V)$, $I_1(V_1)$ и $q(V,V_1)$ обладают следующими свойствами:

$$I(0) = I_1(0) = 0,\ \frac{dI}{dV} > 0,\ \frac{dI_1}{dV_1} > 0,\ \frac{\partial q}{\partial V} > 0,\ \frac{\partial q}{\partial V_1} \le 0. \qquad (2.4)$$

По физическому смыслу все фазовые переменные системы (V, V_1, M, M_1) неотрицательны, поэтому фазовым пространством системы (2.3) является $\mathbf{R}_+^4$ (неотрицательный ортант четырехмерного пространства).

В системе (2.3) первые два уравнения (для V и V_1) образуют замкнутую подсистему с фазовым пространством $\mathbf{R}_+^2$ (первый квадрант плоскости):

$$\frac{dV}{dt} = Q - I - q, \quad \frac{dV_1}{dt} = q - I_1. \qquad (2.5)$$

Состояния равновесия $(V = V^*, V_1 = V_1^*)$ подсистемы (2.5) ищутся как решения системы уравнений (в общем случае трансцендентных):

$$I(V) + q(V,V_1) = Q,$$
$$I_1(V_1) - q(V,V_1) = 0,$$

которая эквивалентна следующей системе (вместо второго уравнения напишем сумму обоих уравнений):

$$I(V) + q(V,V_1) = Q,$$
$$I(V) + I_1(V_1) = Q. \qquad (2.6)$$

Первое уравнение системы (2.6) определяет некоторую неявную функцию $V_1 = f_1(V)$, а второе уравнение – неявную функцию $V_1 = f_2(V)$.

Покажем, что состояние равновесия подсистемы (2.5) *существует и единственно*. Дифференцируя второе уравнение (2.6) по V, имеем

$$\frac{dI}{dV} + \frac{dI_1}{dV_1}\frac{dV_1}{dV} = 0,$$

и поэтому, в силу (2.4), $dV_1 / dV < 0$ для функции $V_1 = f_1(V)$, т.е. эта функция *убывающая*. Аналогично, из первого уравнения (2.6) имеем

$$\frac{dI}{dV} + \frac{\partial q}{\partial V} + \frac{\partial q}{\partial V_1}\frac{dV_1}{dV} = 0,$$

и тогда $dV_1 / dV > 0$ для функции $V_1 = f_2(V)$, т.е. эта функция *возрастающая*. Следовательно, графики функций $V_1 = f_1(V)$ и $V_1 = f_2(V)$ могут пересекаться *не более чем в одной точке*. Будем считать, что $f_1(0) \geq f_2(0)$, где $V_1 = f_1(0)$ — корень уравнения $I_1(V_1) = Q$, а $V_1 = f_2(0)$ — корень уравнения $q(0, V_1) = Q$. Тогда у графиков функций $V_1 = f_1(V)$ и $V_1 = f_2(V)$ *существует единственная* точка пересечения $(V = V^*, V_1 = V_1^*)$, а следовательно, состояние равновесия подсистемы (2.5) существует и единственно.

Теперь покажем, что состояние равновесия $(V = V^*, V_1 = V_1^*)$ *глобально асимптотически устойчиво*, т.е. при любых начальных условиях из $\mathbf{R}_+^2$ траектории подсистемы (2.5) стремятся к состоянию равновесия $(V = V^*, V_1 = V_1^*)$. Для этого покажем, что: 1) состояние равновесия *локально* (из малой окрестности) асимптотически устойчиво; 2) в системе не имеется предельных циклов (а также замкнутых контуров, составленных из траекторий).

Локальная асимптотическая устойчивость состояния равновесия проверяется по первому приближению. Сделаем замену переменных: $\xi = V - V^*$, $\eta = V_1 - V_1^*$; тогда $\dot{\xi} = \dot{V}$ и $\dot{\eta} = \dot{V}_1$, а состояние равновесия в переменных (ξ, η) имеет координаты (0,0). Далее в новых переменных разложим правые части системы (2.5) в ряд Тейлора по переменным ξ и η в окрестности точки (0,0); получим следующую *линеаризованную* систему:

$$\left.\begin{aligned}\dot{\xi} &= \left(-\frac{\partial q}{\partial V} - \frac{dI}{dV}\right)\xi - \frac{\partial q}{\partial V_1}\eta, \\ \dot{\eta} &= \frac{\partial q}{\partial V}\xi + \left(\frac{\partial q}{\partial V_1} - \frac{dI_1}{dV_1}\right)\eta,\end{aligned}\right\} \qquad (2.7)$$

где производные вычисляются в точке $(V = V^*, V_1 = V_1^*)$.

Устойчивость или неустойчивость состояния равновесия (0,0) системы (2.7) (а следовательно, и системы (2.5)) определяется корнями характеристического уравнения $\lambda^2 + A\lambda + B = 0$. Устойчивость имеет место, если все корни этого уравнения лежат слева от мнимой оси; для этого необходимо и достаточно, чтобы было $A > 0$ и $B > 0$. Вычисляя коэффициенты A и B непосредственно, имеем (в силу условий (2.4)):

$$A = \frac{\partial q}{\partial V} + \frac{dI}{dV} - \frac{\partial q}{\partial V_1} + \frac{dI_1}{dV_1} > 0,$$

$$B = -\frac{dI}{dV}\frac{\partial q}{\partial V_1} + \frac{\partial q}{\partial V}\frac{dI_1}{dV_1} + \frac{dI}{dV}\frac{dI_1}{dV_1} > 0.$$

Следовательно, состояние равновесия локально асимптотически устойчиво.

На плоскости из *локальной* асимптотической устойчивости состояния равновесия *не следует* его *глобальная* устойчивость: в общем случае могут существовать также устойчивые *периодические движения*. Но в системе (2.5) таких движений нет по **критерию Бендиксона**. Согласно критерию Бендиксона, если для двумерной динамической системы

$$\dot{V} = F(V, V_1),$$

$$\dot{V}_1 = F_1(V, V_1)$$

выражение $\frac{\partial F}{\partial V} + \frac{\partial F_1}{\partial V_1}$ сохраняет знак и не обращается тождественно в нуль в некоторой области плоскости (V, V_1), то в этой области у системы нет периодических движений (в том числе предельных циклов) и замкнутых контуров, составленных из тра-

екторий. Для системы (2.5), в силу сделанных предположений, имеем:

$$\frac{\partial F}{\partial V}+\frac{\partial F_1}{\partial V_1}=-\frac{dI}{dV}-\frac{\partial q}{\partial V}+\frac{\partial q}{\partial V_1}-\frac{dI_1}{dV_1}<0$$

во всём первом квадранте плоскости (V,V_1). Следовательно, система (2.5) не имеет периодических движений и состояние равновесия $(V=V^*,V_1=V_1^*)$ глобально асимптотически устойчиво. Таким образом, при любых начальных уровнях воды в море и заливе с течением времени обязательно установится водное равновесие.

Процесс установления водного равновесия значительно *более быстрый*, чем установление равновесной концентрации загрязняющих веществ, поэтому рассмотренная подсистема (2.5) – это не только замкнутая, но и *«быстрая»* подсистема.

После установления равновесия в «быстрой» подсистеме можно перейти к рассмотрению *«медленной»* подсистемы (подсистема уравнений для M и M_1 в системе (2.3) при условии $V\equiv V^*$ и $V_1\equiv V_1^*$). Согласно этой подсистеме, равновесная концентрация солей в водоеме $\mu^*=M^*/V^*$ удовлетворяет уравнению (условие обращения в нуль правой части уравнения для dM/dt):

$$\mu^* q(V^*,V_1^*)-\nu Q=0,$$

откуда находим равновесную концентрацию растворимых солей и примесей в водоеме:

$$\mu^*=\frac{\nu Q}{q(V^*,V_1^*)}. \qquad (2.8)$$

Значение $\mu=\mu^*$ является *устойчивым* состоянием равновесия для μ, так как в силу уравнения для dM/dt в (2.3) величина dM/dt (а следовательно, и $d\mu/dt$, так как мы рассматриваем «медленную» подсистему при условии $\mu=M/V^*$) будет положительной при $\mu<\mu^*$ и отрицательной при $\mu>\mu^*$.

Так как равновесные объемы V^* и V_1^* ищутся как решения системы (2.6), то из (2.8) получаем:

$$\mu^* = \frac{\nu Q}{q(V^*, V_1^*)} = \nu \frac{q^* + I^*}{q^*} = \nu \cdot \left(1 + \frac{I^*}{q^*}\right) = \nu \cdot \left(1 + \frac{I^*}{I_1^*}\right), \qquad (2.9)$$

где звездочки обозначают, что соответствующие величины взяты при равновесных значениях $V \equiv V^*$ и $V_1 \equiv V_1^*$.

Отношение I^* / I_1^*, входящее в (2.9), грубо приближенно равно отношению площадей водных поверхностей водоема и залива, т.е. S / S_1. Для Каспийского моря и Кара-Богаз-Гола это отношение примерно равно 37, а следовательно, равновесная концентрация в заливе примерно только в 40 раз больше средней концентрации солей в потоках пресной воды, поступающей в море из рек и с дождями.

Таким образом, залив, подобный Кара-Богаз-Голу, играет роль отстойника, отсасывающего растворимые загрязнения из водоема. Эту же роль в заселяемых стоками воды водоемах могут сыграть искусственные заливы. При этом необходимо обеспечить наличие неисчезающего перепада уровня воды от водоема к заливу, так как обратный поток из залива в водоем сведет на нет отсасывающее действие залива.

2.3.2. Реализация в AnyLogic

Работа реализована в файле `Part1\Caspian Sea.alp` (рис. 2.7). В окне анимации имеются осциллограммы изменения массы соли и уровня воды в водоеме и заливе. На картинке, схематично изображающей Каспийское море и залив Кара-Богаз-Гол, можно наблюдать подъем и опускание уровней воды. Параметры системы (коэффициенты, входящие в состав функций интенсивностей испарения и притока воды, а также концентрацию соли в речных водах) можно изменять с помощью бегунков.

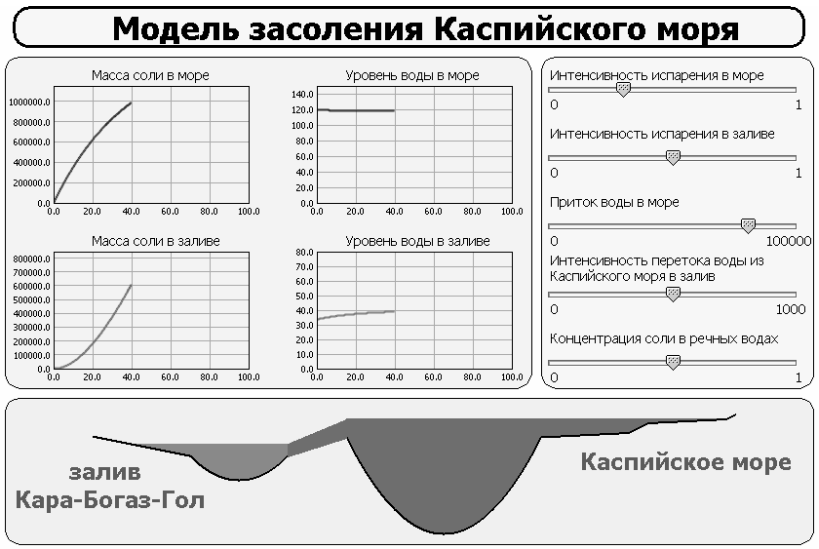

Рис. 2.7

2.3.3. Задания для эксперимента

Изменяя параметры функций интенсивности испарения и притока воды, проследите за изменением динамики системы. Что будет, если искусственно «отрезать» залив от моря с помощью дамбы[3] (обращение в нуль коэффициента интенсивности перетекания воды из моря в залив)?

[3] Именно так было сделано в 1980-е гг. в отношении Каспия и Кара-Богаз-Гола из опасения, что Каспий обмелеет.

3. Классические эффекты в динамике линейных осцилляторов

3.1. Динамика линейного осциллятора

3.1.1. Модель

Математическая модель линейного осциллятора представляет собой линейное дифференциальное уравнение второго порядка:

$$\ddot{x} + 2\delta\dot{x} + \omega^2 x = 0\,. \qquad (3.1)$$

Это уравнение описывает широкий класс процессов в системах различной природы (механических, электрических, экономических и т.д.). При этом параметры модели имеют соответствующий смысл. Например, для физического маятника (грузик на пружине с закрепленным концом, движущийся с вязким трением) за основу берется второй закон Ньютона (равенство для сил): $m\ddot{x} = -h\dot{x} - kx$ (x – координата грузика, m – масса, h – коэффициент трения, k – коэффициент жесткости пружины) или $m\ddot{x} + 2\delta\dot{x} + \omega^2 x = 0$, откуда после деления обеих частей на m получаем уравнение (3.1), где $2\delta = h/m$ и $\omega^2 = k/m$. Для электрического колебательного контура (катушка, резистор и конденсатор) имеет место закон Кирхгофа (равенство для падений напряжения): $L\ddot{q} = -R\dot{q} - q/C$ (q – заряд конденсатора, L – индуктивность катушки, R – сопротивление резистора, C – емкость конденсатора) или $L\ddot{q} + R\dot{q} + q/C = 0$, откуда после деления на L получаем уравнение (3.1), где $2\delta = R/L$ и $\omega^2 = 1/(CL)$.

Уравнение (3.1) задает динамическую систему, состоянием которой является $(x, \dot{x})$. В этой системе имеется состояние равновесия $x = \dot{x} = 0$, которое в зависимости от корней характеристического уравнения $\lambda^2 + 2\delta\lambda + \omega^2 = 0$ может иметь различный тип. В свою очередь, характеристические корни $\lambda_{1,2}$ зависят от параметров δ и ω^2.

Если на осциллятор действует внешнее гармоническое возмущение $F_0 e^{i\nu t}$ (в механическом случае – сила, в электрическом – ЭДС), то система описывается линейным неоднородным дифференциальным уравнением:

$$\ddot{x} + 2\delta\dot{x} + \omega^2 x = F_0 e^{i\nu t} . \qquad (3.2)$$

При $\delta > 0$ (положительное трение) собственные колебания осциллятора затухают, и устанавливаются вынужденные колебания, которые можно записать в виде $x(t) = F_0 K(i\nu) e^{i\nu t}$ или $x(t) = F_0 A(\nu) e^{i[\nu t + \varphi(\nu)]}$. Величина $K(i\nu) = A(\nu) e^{i\varphi(\nu)}$ (комплексный коэффициент усиления амплитуды внешнего воздействия) называется **амплитудно-фазовой частотной характеристикой (АФЧХ)** и графически изображается в виде **годографа** на комплексной плоскости (ν изменяется от 0 до $+\infty$). Величина $A(\nu)$ (отношение амплитуды вынужденных колебаний к амплитуде внешнего воздействия) называется **амплитудно-частотной характеристикой (АЧХ)**, а величина $\varphi(\nu)$ (фазовый сдвиг вынужденных колебаний по отношению к вынуждающим) – **фазово-частотной характеристикой (ФЧХ)**.

При определенном значении внешней частоты $\nu = \nu^*$, зависящем от параметров осциллятора, значение АЧХ достигает максимума, т.е. вынужденные колебания происходят с наибольшей амплитудой. Это явление называется **резонансом**, а само значение $\nu = \nu^*$ – **резонансной частотой**.

3.1.2. Реализация в AnyLogic

Работа реализована в файле `Part2\Oscillator.alp` (рис. 3.1). В окне анимации с помощью радиокнопки из группы радиокнопок «Колебания» можно выбирать тип исследуемых колебаний (собственные или вынужденные).

В режиме «Собственные колебания» в левой части окна рисуется бифуркационная диаграмма на плоскости параметров (ω^2, δ), на которой с помощью точки (маленького прямоуголь-

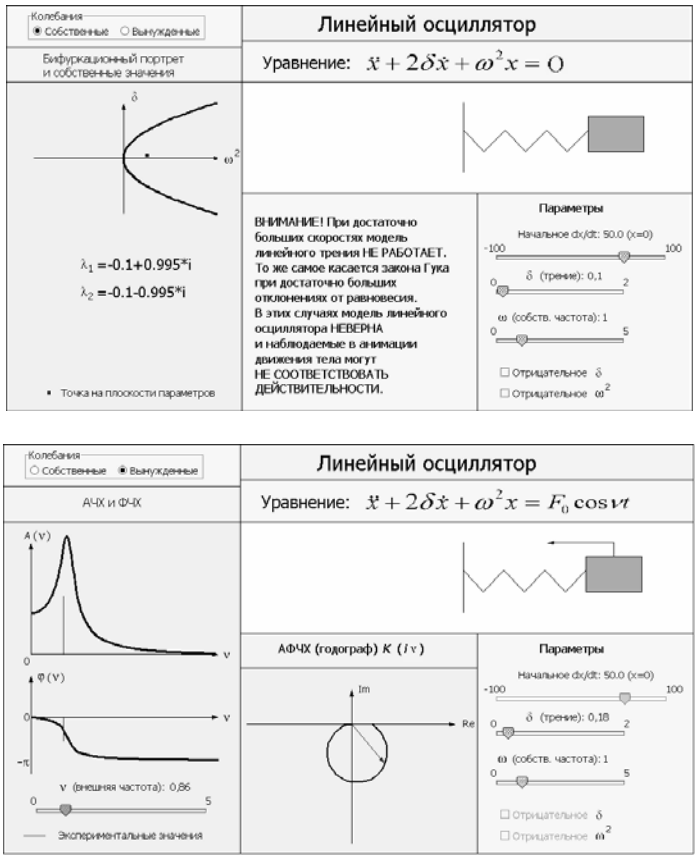

Рис. 3.1

ника) выводятся текущие значения параметров, под диаграммой – значения собственных чисел (характеристических корней) $\lambda_{1,2}$. В правой части окна – анимация колеблющегося грузика на пружине с возможностью изменения параметров δ, ω^2 и начальной скорости dx/dt (начальное значение x при изменении dx/dt автоматически делается равным нулю). С помощью флажков можно изменять знак параметров δ и ω^2.

В режиме «Вынужденные колебания» выводятся графики АЧХ и ФЧХ, а также годограф АФЧХ (при этом значение АФЧХ изображено в виде радиус-вектора). При этом, кроме теоретических графиков АЧХ и ФЧХ, выводятся экспериментальные значения этих характеристик, вычисляемые в каждый момент времени по формуле:

$$A(\nu) = \begin{cases} \sqrt{x^2 + \dot{x}^2 / \nu^2} / F_0 & \text{при } \nu \neq 0; \\ x / F_0 & \text{при } \nu = 0; \end{cases};$$

$$\varphi(\nu) = \begin{cases} \text{arctg}\,(-Y / X) & \text{при } X > 0; \\ \left[\pi - \text{arctg}\,(-Y / X)\right] \cdot \text{sgn}\,(-Y) & \text{при } X \leq 0, \end{cases}$$

где $X(t) = \int_0^t x(\tau) \cos x(\tau)\, d\tau$, $Y(t) = \int_0^t x(\tau) \sin x(\tau)\, d\tau$.

В правой части окна имеется анимация колеблющегося грузика на пружине, на который действует внешняя гармоническая сила, показанная стрелкой (длина стрелки указывает модуль силы, а направление стрелки соответствует направлению силы). С помощью бегунков можно менять параметры осциллятора δ и ω^2, начальное значение скорости dx/dt (начальное значение x при изменении dx/dt автоматически делается равным нулю), а также (под графиками АЧХ и ФЧХ) частоту внешнего воздействия ν. Флажки изменения знаков параметров δ и ω^2 в этом режиме недоступны (так как АФЧХ не имеет смысла при отрицательном δ или ω^2 из-за неустойчивости собственных колебаний).

3.1.3. Задания для эксперимента

1. Запустите модель в AnyLogic.

2. Пронаблюдайте за эволюцией типов собственных движений грузика в анимации, а также фазового портрета и осциллограмм фазовых переменных при изменении параметров осциллятора. Согласуется ли видимое вами на экране с теорией?

3. Выберите значения параметров, при которых собственные колебания затухают. Затем выберите с помощью флажка режим «Вынужденные колебания». Как изменяются годограф АФЧХ и графики АЧХ и ФЧХ (в том числе резонансная частота и значение АЧХ в резонансе) при изменении коэффициента трения? при изменении собственной частоты? Объясните наблюдаемые эффекты теоретически.

4. Сходятся ли экспериментальные значения АЧХ и ФЧХ к теоретическим с течением времени?

3.2. Динамический демпфер

3.2.1. Модель

Динамический демпфер – конструкция, предназначенная для ослабления (в идеале – подавления) нежелательных вибраций тела на упругом основании, возникающих при действии внешней гармонической силы на это тело (будем в дальнейшем называть его **основным телом**). Примером такого тела является автомобиль с рессорами, движущийся по дороге, на которой много рытвин, или станок, имеющий эксцентричные вращающиеся детали. Динамический демпфер представляет собой еще одно тело, упруго скрепленное с основным.

Физическая модель рассматриваемой системы состоит из основного тела массы M, соединенного с неподвижным основанием пружиной жесткости K (или несколькими пружинами общей жесткости K для повышения надежности скрепления), и демпфирующего тела (демпфера) массы m, скрепленного пружиной жесткости k с основным телом (рис. 3.2).

Пусть на основное тело действует комплексная внешняя гармоническая сила $F_0 e^{i\nu t}$ ($F_0 > 0$). Пусть x – отклонение основного тела от положения, в котором пружина K недеформирована, а y – отклонение демпфера от положения, в котором пружина k недеформирована. Пусть в системе действует сила вязкого трения с коэффициентом h. Тогда математическая модель движений ос-

новного тела с демпфером представляет собой систему дифференциальных уравнений:

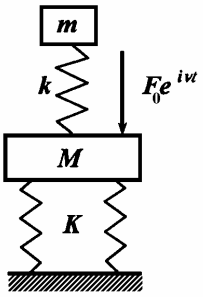

Рис. 3.2

$$\begin{aligned} M\ddot{x} &= -Kx - h(\dot{x} - \dot{y}) - k(x - y) + F_0 e^{i\nu t}, \\ m\ddot{y} &= h(\dot{x} - \dot{y}) + k(x - y). \end{aligned} \tag{3.3}$$

Будем искать вынужденные колебания системы (3.2) в виде: $x = Ae^{i\nu t}$, $y = Be^{i\nu t}$. Подставим эти функции в (3.3). После сокращения на $e^{i\nu t}$ имеем:

$$\begin{aligned} -M\nu^2 A &= -KA - (hi\nu + k)(A - B) + F_0, \\ -m\nu^2 B &= (hi\nu + k)(A - B), \end{aligned}$$

откуда получаем:

$$A = \frac{F_0 \cdot (hi\nu + k - m\nu^2)}{(K + hi\nu + k - M\nu^2)(hi\nu + k - m\nu^2) - (hi\nu + k)^2},$$

$$B = \frac{F_0 \cdot (hi\nu + k)}{(K + hi\nu + k - M\nu^2)(hi\nu + k - m\nu^2) - (hi\nu + k)^2}.$$

Нас интересует *вещественная* амплитуда вынужденных колебаний основного тела:

$$|A| = F_0 \cdot \left\{ \left[k(K - M\nu^2) - m\nu^2 (K + k - M\nu^2) \right]^2 + \right.$$
$$\left. + h^2\nu^2 \left[K - (M + m)\nu^2 \right]^2 \right\}^{-1/2} \cdot \left[(k - m\nu^2)^2 + h^2\nu^2 \right]^{1/2}.$$

Отсюда следует, что величину $|A|$ *нельзя обратить в нуль* за счет подбора параметров демпфера (*k* и *m*) из-за наличия трения ($h \neq 0$). Следовательно, при наличии трения можно добиться только некоторого положительного минимума $|A|$. Если трение исчезающе мало, то выбор параметров из условия $k/m = \nu^2$ (настройка параметров демпфера *в резонанс* по отношению к внешнему воздействию при условии, что разность фаз колебаний демпфера и внешнего воздействия близка к π, т.е. к *противофазе*) приведет к достаточно малой величине $|A|$, т.е. к очень малым вибрациям основного тела. В идеальном случае, когда трение отсутствует ($h = 0$), можно выбрать *k* и *m* из условия $k/m = \nu^2$ и тогда *формально* будет $|A| = 0$ (колебания основного тела будут подавлены полностью); но *фактически* при $h = 0$ *нельзя говорить о вынужденных колебаниях*, так как переходный процесс *не затухает* и вынужденные колебания *не устанавливаются*.

Как и для одиночного осциллятора с внешним гармоническим воздействием, в системе с динамическим демпфером для основного тела можно ввести амплитудно-частотную характеристику (АЧХ) как отношение $|A|/F_0$.

3.2.2. Реализация в AnyLogic

Работа реализована в файле `Part2\Dynamic Damper.alp` (рис. 3.3). В окне анимации можно наблюдать колеблющееся основное тело без динамического демпфера или с динамическим демпфером (демпфер подключается и отключается с помощью флажка «Подсоединить демпфер»). При этом внешняя гармоническая сила, действующая на основное тело, показана стрелкой, у которой меняются длина и направление (как и в работе «Динамика линейного осциллятора», см. п. 3.1.2).

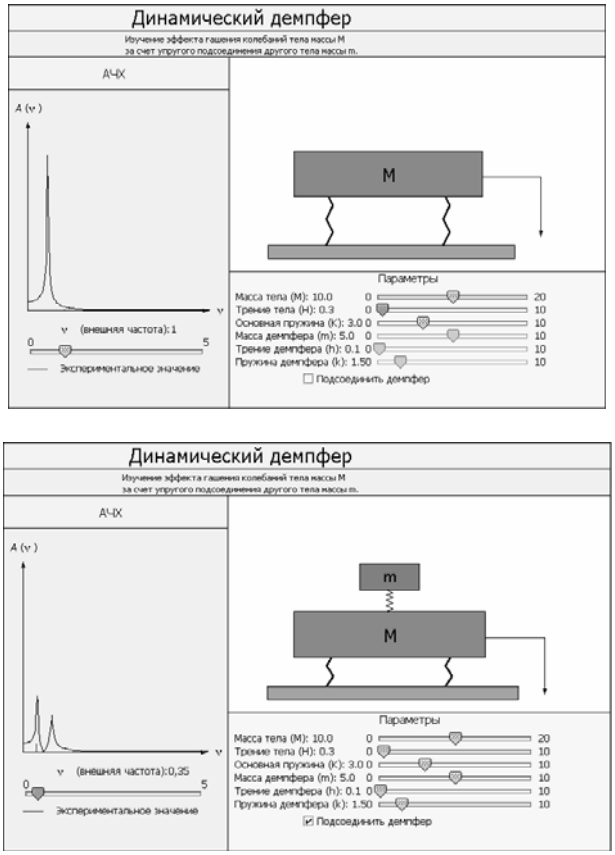

Рис. 3.3

3.2.3. Задания для эксперимента

1. В режиме колебаний без динамического демпфера проследите за возможными видами собственных колебаний основного тела. Как изменяется АЧХ при изменении параметров основного тела и частоты внешнего воздействия?

2. Подключите динамический демпфер, зафиксируйте массу основного тела и попытайтесь при различных значениях внешней частоты ν добиться минимизации амплитуды колебаний основного тела, изменяя только параметры динамического демпфера. Какие параметры и каким образом нужно для этого изменять?

3. Сходится ли экспериментальное значение АЧХ к теоретическому с течением времени?

3.3. Связанные осцилляторы

3.3.1. Модель

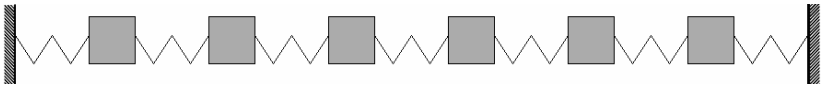

Рис. 3.4

Рассмотрим систему из n упруго связанных линейных осцилляторов (рис. 3.4). Пусть m_i – масса i-го осциллятора, x_i – отклонение i-го осциллятора от положения равновесия (положение, в котором все пружины недеформированы) ($i = \overline{1,n}$; направление отсчета x_i выберем одним и тем же для всех i). Пусть $\mu_{i,\,i+1}$ – жесткость пружины, соединяющей i-й и $(i+1)$-й осцилляторы ($i = \overline{1,n-1}$), k_1 и k_2 – жесткости крайних пружин. Будем считать, что вязкое трение в системе достаточно мало, так что можно им пренебречь. Тогда математическая модель колебаний системы имеет вид:

$$\left.\begin{aligned} &m_1\ddot{x}_1 = -k_1x_1 - \mu_{12}(x_1 - x_2);\\ &m_i\ddot{x}_i = -\mu_{i-1,\,i}(x_i - x_{i-1}) -\\ &-\mu_{i,\,i+1}(x_i - x_{i+1})(1 < i < n);\\ &m_n\ddot{x}_n = -k_nx_n - \mu_{n-1,\,n}(x_n - x_{n-1}). \end{aligned}\right\} \qquad (3.4)$$

Система (3.4) является линейной однородной, поэтому ее решение можно искать в виде:

$$x_i = a_i e^{\lambda t} \ (i = \overline{1,n}). \qquad (3.5)$$

Значения λ находятся из характеристического уравнения системы (3.4). Характеристическое уравнение получается приравниванием нулю определителя линейной системы для коэффициентов a_i, которая получается подстановкой (3.5) в (3.4) с последующим сокращением на $e^{\lambda t}$. Если какая-либо пара комплексно сопряженных значений λ равна $\pm i\omega$, где ω – вещественное положительное число, то значение ω называется **нормальной частотой** системы осцилляторов, а соответствующее базисное решение – **нормальным колебанием** с частотой ω. Нормальное колебание с частотой ω означает, что все осцилляторы колеблются *с одной и той же частотой* ω. Из базисных решений $x_i = a_i e^{\lambda t}$ путем взятия линейной комбинации с произвольными коэффициентами получается общее решение системы (3.4), дающее большое разнообразие движений связанных осцилляторов. В общем случае движения системы могут быть **периодическими** (нормальные частоты находятся в *рациональном* отношении) или **квазипериодическими** (нормальные частоты находятся в *иррациональном* отношении).

При $n = 2$ могут иметь место **биения** – поочередная перекачка энергии от одного осциллятора к другому, причем она может быть полной или неполной. В частности, при $m_1 = m_2 = m$, $k_1 = k_2 = k$ и $\mu_{12} = \mu$ характеристическое уравнение системы (3.4) имеет следующие корни:

$$\lambda_{1,2} = \pm i\sqrt{\frac{k}{m}} = \pm i\omega_1, \ \lambda_{3,4} = \pm i\sqrt{\frac{k+2\mu}{m}} = \pm i\omega_2.$$

Корням $\lambda = \lambda_{1,2}$ соответствует (в комплексной форме) частное решение $x_1 = C_1 e^{i\omega_1 t}$, $x_2 = C_1 e^{i\omega_1 t}$ (C_1 — произвольная постоянная), а корням $\lambda = \lambda_{3,4}$ — частное решение $x_1 = C_2 e^{i\omega_2 t}$,

$x_2 = -C_2 e^{i\omega_2 t}$ (C_2 — произвольная постоянная). Решением (не общим решением!) системы (3.4) будет также сумма этих частных решений. Пусть $|C_1| \geq |C_2|$; тогда эту сумму частных решений можно записать в виде:

$$\left.\begin{aligned} x_1 &= C_1 e^{i\omega_1 t}\left[1 + \frac{C_2}{C_1} e^{i(\omega_2-\omega_1)t}\right], \\ x_2 &= C_1 e^{i\omega_1 t}\left[1 - \frac{C_2}{C_1} e^{i(\omega_2-\omega_1)t}\right]. \end{aligned}\right\} \qquad (3.6)$$

Каждое из колебаний x_1 и x_2 в (3.6) можно трактовать как колебание с частотой ω_1 (**несущая частота**); его амплитуда $\left|1 \pm \frac{C_2}{C_1} e^{i(\omega_2-\omega_1)t}\right|$ гармонически меняется во времени с частотой $\omega_2 - \omega_1$ (**частота модуляции**). Если жесткость связывающей пружины μ мала, то $\omega_1 \approx \omega_2$ и амплитуды колебаний (3.6) являются *медленно меняющимися функциями времени*, которые меняются в пределах от $|C_1| \cdot \left(1 - |C_2 / C_1|\right)$ до $|C_1| \cdot \left(1 + |C_2 / C_1|\right)$. В результате имеем движения осцилляторов с поочередной перекачкой энергии, причем при $|C_1| = |C_2|$ эта перекачка будет *полной* (один из осцилляторов перестает колебаться, в то время как другой колеблется с максимальной амплитудой) (рис. 3.5).

При $|C_1| \leq |C_2|$ получаем аналогичные результаты, только теперь мы имеем колебания системы с частотой ω_2 и амплитудой, меняющейся во времени (при малом μ – медленно меняющейся) в пределах от $|C_2| \cdot \left(1 - |C_1 / C_2|\right)$ до $|C_2| \cdot \left(1 + |C_1 / C_2|\right)$.

В общем случае (при произвольном *n* и не очень малых коэффициентах жесткости связывающих пружин $\mu_{i,\, i+1}$) также имеют место перекачки энергии (более сложного характера) между ос-

цилляторами, причем частота перекачек зависит главным образом от коэффициентов $\mu_{i,\,i+1}$.

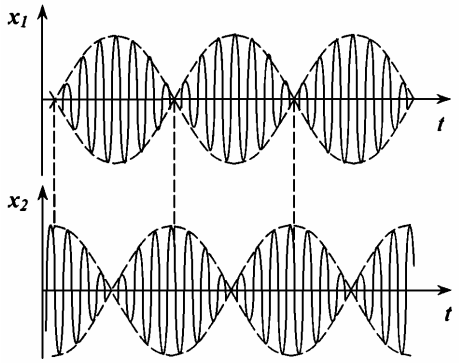

Рис. 3.5

При достаточно больших n (при условии относительной малости расстояний между равновесными положениями осцилляторов) *дискретная* система связанных осцилляторов хорошо аппроксимирует *распределенную* упругую среду (однородную или неоднородную в зависимости от соотношений параметров связанных осцилляторов), и колебания системы близки к одномерным волновым процессам [1]. Можно показать, что с помощью предельного перехода ($n \to \infty$) уравнения движения системы из n связанных осцилляторов превращаются в одномерное волновое уравнение (см., например, [4]).

3.3.2. Реализация в AnyLogic

Реализованы три различные работы (файлы находятся в папке `Part2)`: «Два связанных осциллятора» (файл `2 Coupled Oscillators.alp`, рис. 3.6), «Три связанных осциллятора» (файл `3 Coupled Oscillators.alp`, рис. 3.7) и «Связанные осцилляторы и волновые процессы» (файл `Coupled Oscilla-`

tors and Waves.alp, рис. 3.8). В каждой из работ в окне анимации можно наблюдать колебания различных систем связанных осцилляторов, изменяя параметры осцилляторов и начальные условия.

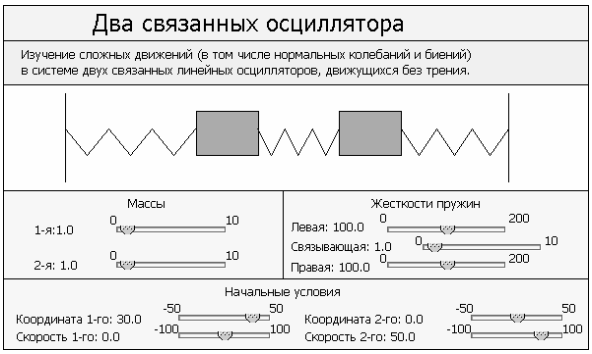

Рис. 3.6

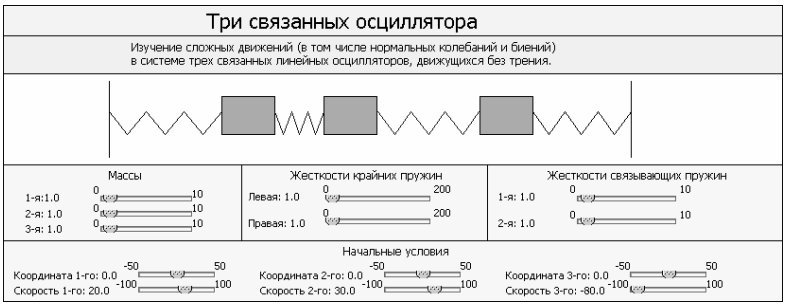

Рис. 3.7

В работе «Связанные осцилляторы и волновые процессы», в которой взято достаточно большое число осцилляторов (в данном случае – шесть), можно наблюдать аппроксимацию волновых процессов колебаниями связанных осцилляторов. Для этого в отдельной области окна анимации показан «профиль волны» (ди-

намика отклонений каждого осциллятора от состояния равновесия).

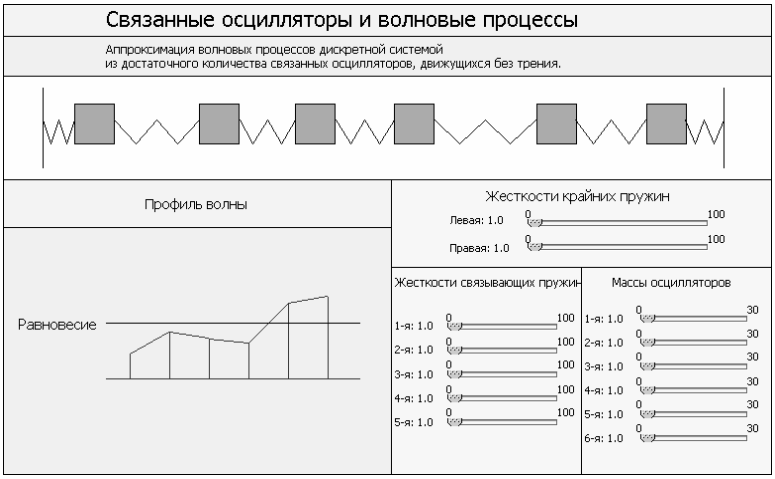

Рис. 3.8

3.3.3. Задания для эксперимента

1. Для систем двух и трех связанных осцилляторов, изменяя их параметры и начальные условия, сделайте выводы об изменении характера нормальных колебаний от собственных параметров каждого осциллятора и от коэффициентов жесткости связывающих пружин.

2. Для систем двух и трех связанных осцилляторов, изменяя их параметры, сделайте выводы об изменении характера перекачек энергии при изменении коэффициентов жесткости связывающих пружин.

3. В работе «Связанные осцилляторы и волновые процессы», изменяя параметры осцилляторов, сделайте выводы о влиянии этих параметров на профиль волны, в частности на период колебаний и длину волны.

4. Эффекты в нелинейных динамических системах

4.1. Динамика электрической схемы с неоновой лампой

4.1.1. Модель

Следуя [2], рассмотрим электрическую схему с неоновой лампой (рис. 4.1).

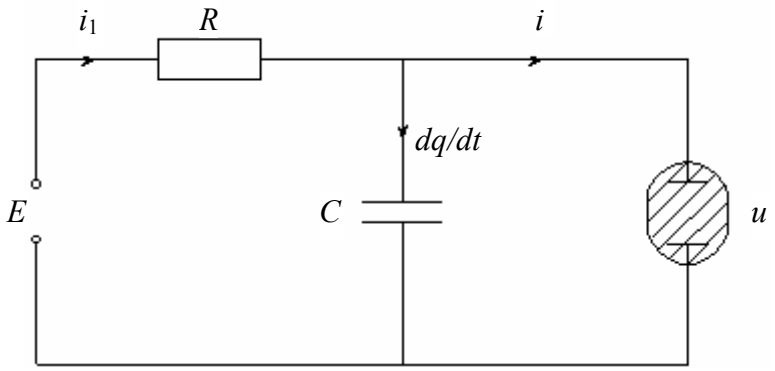

Рис. 4.1

Эта схема состоит из колебательного контура (конденсатор C, резистор R и источник постоянного тока с ЭДС $E = \text{const}$) и неоновой лампы, включенной параллельно конденсатору. Если неоновая лампа пропускает ток, то газ, наполняющий ее, находится в ионизированном состоянии и светится – лампа горит; если она не пропускает ток, то не горит. Неоновую лампу можно считать нелинейным сопротивлением, у которого зависимость между током (силой тока) i и напряжением u (**вольт-амперная характеристика**) выражается функцией $i = f(u)$. График этой функции представлен на рис. 4.2.

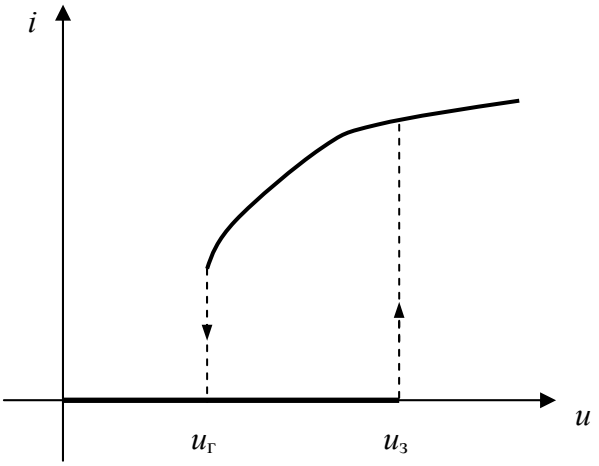

Рис. 4.2

Особенностью функции $i = f(u)$ является ее **гистерезисный** (запаздывающий) характер: если лампа не горела ($i = 0$), то при увеличении напряжения u она загорается (сила тока меняется скачком от нуля до некоторого положительного значения) только при переходе u через некоторое пороговое значение $u_з$ (напряжение зажигания); если же лампа горела, то при уменьшении u она погаснет (ток падает скачком до нуля) только при переходе u через *другое* пороговое значение $u_г$ (напряжение гашения), где $u_г < u_з$. Поэтому на интервале $u_г < u < u_з$ функция $i = f(u)$ *двузначная*, и ее значение зависит не только от текущего значения u, но и от предыстории изменения u.

Применяя законы Кирхгофа для токов и учитывая, что $i = \dot{q}$, где q – заряд на конденсаторе, имеем систему уравнений:

$$i_1 = i + \dot{q}, \quad R i_1 + u = E, \quad i = f(u).$$

Исключая из этой системы i_1, i и q, получим уравнение для u:

$$\dot{u} = \frac{1}{RC} \cdot \left[E - u - R f(u) \right]. \qquad (4.1)$$

Это уравнение не определяет динамическую систему (а переменная u не является состоянием динамической системы), так как

на интервале $u_г < u < u_з$ правая часть уравнения не определена из-за двузначности функции $f(u)$. Поэтому для полного описания динамики системы нужно, кроме переменной *u*, ввести логическую переменную («флажок») æ (каппа) (индикатор горения лампы): если лампа горит, то $æ = 1$, а если не горит, то $æ = 0$. При переходе *u* в сторону увеличения через значение $u = u_з$ значение æ скачком изменяется с 0 на 1, а при переходе *u* в сторону уменьшения через значение $u = u_г$ – с 1 на 0. Состоянием (фазовыми переменными) динамической системы является вектор $(u, æ)$.

В зависимости от значения параметров *E*, *R* и *C* может быть 4 варианта фазового портрета данной динамической системы:

1) При любых начальных условиях лампа обязательно *загорится* и *далее будет гореть* (единственное глобально устойчивое *состояние равновесия* при $æ = 1$).

2) При любых начальных условиях лампа обязательно *погаснет* и *больше не загорится* (единственное глобально устойчивое *состояние равновесия* при $æ = 0$).

3) При любых начальных условиях лампа будет *мигать* (глобально устойчивые *автоколебания*).

4) Существуют *два* устойчивых состояния равновесия, в одном из которых лампа горит, а в другом – не горит. Траектория системы приходит в одно из этих состояний равновесия в зависимости от начальных условий. Если под действием некоторых флуктуаций (случайных электрических импульсов) начальное напряжение на лампе изменится скачком, то система может перейти по фазовой траектории в другое состояние равновесия. Такое поведение системы аналогично поведению электрической схемы, называемой **триггером**. Поэтому данный сценарий поведения системы можно назвать **триггерным режимом**.

4.1.2. Реализация в AnyLogic

Работа реализована в файле `Part3\neon.alp` (рис. 4.3). В окне анимации выведены уравнения модели, осциллограмма из-

менения $u(t)$, а также анимация прохождения токов и зарядов через систему.

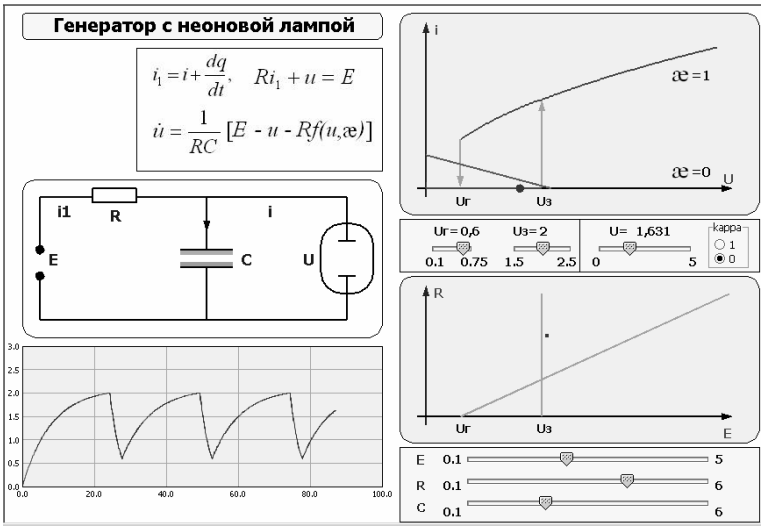

Рис. 4.3

На конденсаторе яркость цветов пластин показывает величину (модуль) заряда, а красный и синий цвета – полярность заряда (при изменении полярности цвета пластин меняются местами). Если ток через лампу идет, то ее внутренность подсвечивается красным цветом (имитация газового разряда), а если не идет, то внутренность лампы обесцвечивается (газовый разряд прекращается). На отдельной фазовой плоскости (u, i) по фазовым траекториям движется точка. Текущее значение æ показывает выбранная радиокнопка из группы радиокнопок «`kappa`».

Кроме того, имеется бифуркационная диаграмма на плоскости параметров (E, R).

С помощью бегунков можно изменять параметры системы, а также значения напряжений зажигания и гашения и начальные условия для u и æ.

4.1.3. Задания для эксперимента

1. Выбирая различные начальные условия и значения параметров, проследите за бифуркациями (качественными изменениями структуры) фазового портрета. Исследуйте отдельно триггерный режим, изменяя начальное значение *u*.

2. Выберите значения параметров так, чтобы они попали в область на плоскости (E, R), соответствующую возбуждению автоколебаний. Экспериментально выясните зависимость периода автоколебаний от параметров, постройте соответствующие графики.

4.2. Часы Галилея – Гюйгенса

4.2.1. Модель

Математическая модель малых колебаний обычного маятника с учетом вязкого трения – это модель линейного осциллятора:

$$\ddot{\varphi} + 2\delta\dot{\varphi} + \omega^2\varphi = 0\,, \qquad (4.2)$$

где φ – угол отклонения маятника от вертикали, δ – коэффициент вязкого трения, ω – частота свободных колебаний маятника при отсутствии вязкого трения ($\omega^2 = \frac{g}{l}$, где g – ускорение свободного падения, l – длина нити маятника). Уравнение (4.2) задает оператор динамической системы, состоянием которой (вектором фазовых переменных) является вектор $(\varphi, \dot{\varphi})$. При $\delta = 0$ (отсутствие вязкого трения) маятник совершает свободные *незатухающие* синусоидальные колебания, период которых не зависит от начальных условий (угла φ и угловой скорости $\dot{\varphi}$). Постоянство периода колебаний маятника (при малых отклонениях) было впервые установлено Г. Галилеем.

Однако в реальности вязкое трение всегда присутствует ($\delta > 0$), и решение уравнения (4.2) при небольших δ ($\delta^2 < \omega^2$) имеет вид *затухающих* синусоидальных колебаний с частотой

$\Omega = \sqrt{\omega^2 - \delta^2}$ (при любых начальных условиях фазовые траектории системы стремятся при $t \to +\infty$ к устойчивому состоянию равновесия $(\varphi = 0, \dot{\varphi} = 0)$). Чтобы можно было использовать маятник в качестве *часов*, нужно *считать* его колебания и *показывать* их (например, стрелкой на циферблате). Кроме того, нужно не дать колебаниям маятника затухнуть, т.е. требуется превратить затухающие *свободные колебания* в незатухающие *автоколебания*. Обе эти задачи были решены Х. Гюйгенсом, который предложил устройство, называемое **часовым ходом**. Простейший вариант часового хода изображен на рис. 4.4.

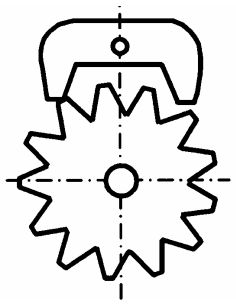

Рис. 4.4

После каждого колебания маятника туда и обратно храповое колесо (храповик) под влиянием заведенной пружины или опускающегося груза поворачивается на один зубец и одновременно сообщает маятнику подталкивающий импульс. Тем самым скорость вращения храпового колеса определяется частотой колебаний маятника, а зубцы храповика в момент его поворота подталкивают маятник, поддерживая его колебания. Таким образом, с помощью часового хода в маятнике реализуется **автоматическое управление** (**обратная связь по состоянию**).

Математическая модель маятника с вязким трением и часовым ходом, сообщающим маятнику мгновенный подталкивающий импульс (удар), имеет вид:

$$\begin{aligned} &\ddot{\varphi} + 2\delta\dot{\varphi} + \omega^2\varphi = 0, \\ &\dot{\varphi}_+ = \dot{\varphi}_- + p, \end{aligned} \qquad (4.3)$$

где $\dot{\varphi}_-$ – доударная угловая скорость, а $\dot{\varphi}_+$ – послеударная (угол φ при этом измениться не успевает). Удар происходит при некотором $\varphi = \alpha$ (в частности, α может быть равно нулю, что соответствует нижнему положению маятника) и $\dot{\varphi} > 0$.

Нарисуем фазовую траекторию, соответствующую одному полному колебанию от значения $\varphi = \alpha$ снова к $\varphi = \alpha$. Пусть M_0 – начальная точка, а M_1 – точка повторного значения $\varphi = \alpha$ при условии, что движение происходит в соответствии с дифференциальным уравнением (4.2). В момент прихода в точку M_1 происходит передача импульса p, и точка M_1 перемещается вдоль оси $\dot{\varphi}$ на расстояние p в точку M_2 (рис. 4.5).

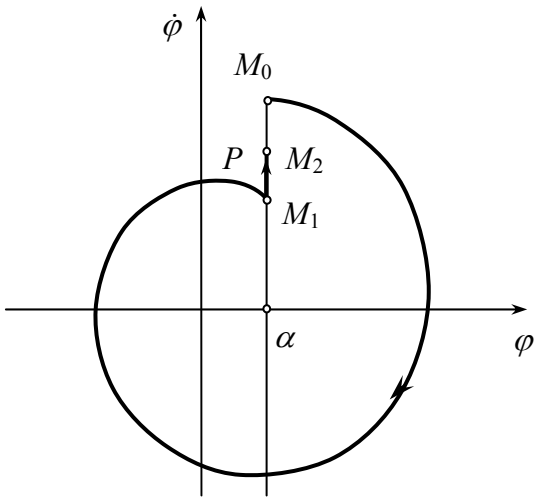

Рис. 4.5

Обозначим через u значение $\dot{\varphi}$ в точке M_0, через $\tilde{u}$ – значение $\dot{\varphi}$ в точке M_1, а через $\bar{u}$ – значение $\dot{\varphi}$ в точке M_2. Тогда, решая дифференциальное уравнение (4.2) при начальных условиях $(\varphi = \alpha, \dot{\varphi} = 0)$ и учитывая, что период одного полного синусоидального колебания равен $2\pi / \Omega$ $(\Omega = \sqrt{\omega^2 - \delta^2})$, получаем:

$$\tilde{u} = e^{-2\pi\delta/\Omega} \cdot u .$$

Так как в точке M_1 происходит передача импульса p, имеем $\bar{u} = \tilde{u} + p$. Отсюда получаем формулу **точечного отображения** (или **функции последования**) – отображения прямой $\varphi = \alpha$ в себя по фазовой траектории динамической системы (4.3); эта формула связывает значения u и $\bar{u}$:

$$\bar{u} = e^{-2\pi\delta/\Omega} \cdot u + p . \qquad (4.4)$$

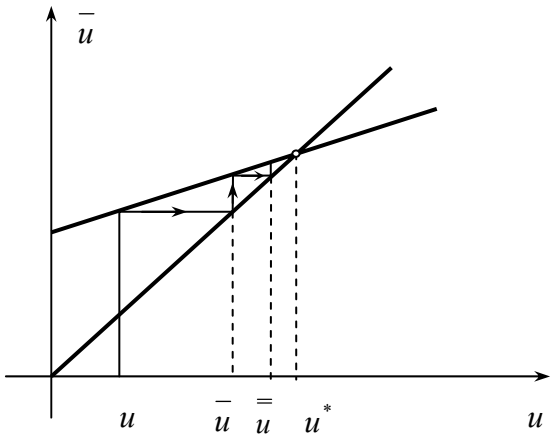

Рис. 4.6

На рис. 4.6 изображена **диаграмма Кёнигса – Ламерея** (или просто **диаграмма Ламерея**), показывающая последовательность значений u, $\bar{u}$, $\bar{\bar{u}}$, ..., получающаяся из начального значения u в силу точечного отображения (4.4) в соответствии с графиком

функции (4.4) и биссектрисой угла (лучом $\bar{u} = u$). Из диаграммы Ламерея видно, что эта последовательность значений стремится к устойчивой **неподвижной точке** (точке, переходящей при отображении (4.4) в себя) u^*, соответствующей автоколебаниям часов. Эта неподвижная точка устойчива *глобально*, т.е. система выходит на автоколебательный режим при любых начальных условиях. Величина u^* находится из уравнения (4.4), если и вместо u, и вместо $\bar{u}$ подставить u^*:

$$u^* = \frac{p}{1 - e^{-2\pi\delta/\Omega}}.$$

4.2.2. Реализация в AnyLogic

Работа реализована в файле `Part3\clock.alp` (рис. 4.7).

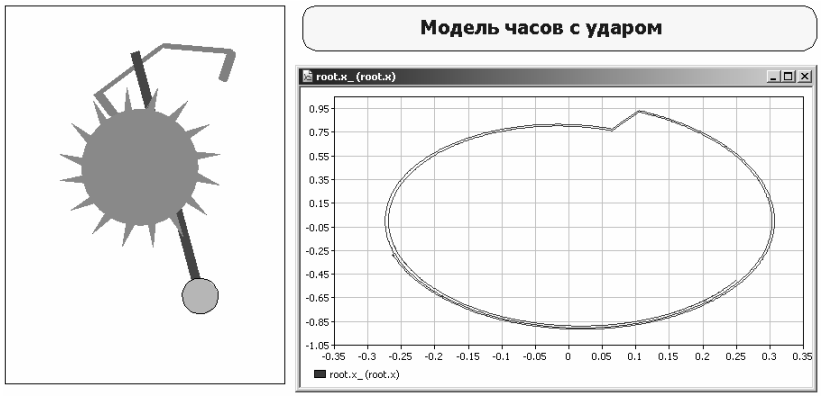

Рис. 4.7

В окне анимации наблюдаются колебания маятниковых часов с часовым ходом. Фазовый портрет на плоскости $(\varphi, \dot{\varphi})$ выводится в отдельном окне, не входящем в состав анимации (для его отображения на экране нужно в окне анимации выбрать вкладку «`root.x_(root.x)`»). Рекомендуется вначале запустить модель на один шаг, затем на панели инструментов AnyLogic вы-

брать из выпадающего списка «*Скорость выполнения модели*» значение «0.1x» и далее запустить модель обычным образом.

4.2.3. Задания для эксперимента

1. Используя счетчик «*Время*» в строке состояний окна AnyLogic, определите период автоколебаний часов.

2. Попробуйте реализовать в AnyLogic модель часов, в которой можно было бы изменять величину подталкивающего импульса p. Запустите эту модель, проследите зависимость периода автоколебаний от значения p и сравните с теоретической зависимостью.

4.3. Ламповый генератор электрических колебаний

4.3.1. Модель

Ламповый генератор является в определенном смысле электрическим аналогом часов Галилея – Гюйгенса: в этой схеме также осуществляется автоматическое управление (обратная связь по состоянию) в виде управляемой подкачки энергии в колебательный контур. Подкачка энергии осуществляется специальным устройством – электронной лампой (радиолампой) или полупроводниковым триодом через взаимоиндукцию. На рис. 4.8 изображена схема лампового генератора. Здесь: L – индуктивность, C – емкость, R – сопротивление (параметры колебательного контура); M – коэффициент взаимной индукции катушек; E_g – напряжение на управляющей сетке лампы (относительно катода).

При подаче напряжения на сетку через лампу потечет анодный ток I_a, величина которого определяется величиной E_g. Экспериментальный график зависимости I_a от E_g (**анодно-сеточная характеристика** лампы) представлен на рис. 4.9.

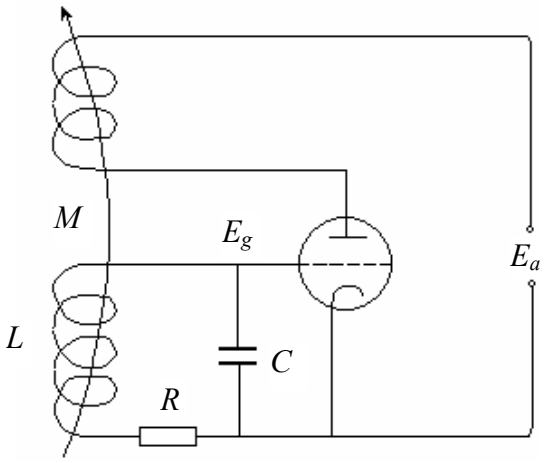

Рис. 4.8

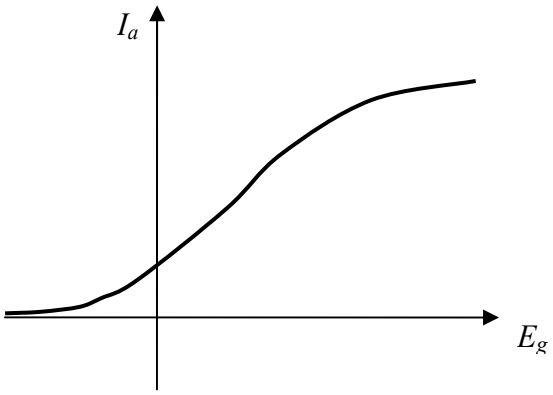

Рис. 4.9

Эту нелинейную зависимость можно аппроксимировать по-разному, например полиномом 3-й степени:

$$I_a = \alpha + \beta E_g - \gamma E_g^{\,3}, \qquad (4.5)$$

где $\alpha, \beta, \gamma > 0$.

Обозначим заряд конденсатора через q, а силу тока в контуре – через I. Для составления математической модели лампового генератора воспользуемся очевидными электродинамическими соотношениями:

$$I = \dot{q}\ ,\ E_g = \frac{q}{C}\,, \tag{4.6}$$

а также законом Кирхгофа для замкнутого контура вместе с законом взаимной индукции:

$$L\frac{dI}{dt} + RI + \frac{q}{C} = M\frac{dI_a}{dt}\,. \tag{4.7}$$

Подставим (4.5) и (4.6) в уравнение (4.7); получим (после деления на L) следующее дифференциальное уравнение для q:

$$\ddot{q} + \left(\frac{R}{L} - \frac{M\beta}{LC}\right)\dot{q} + \frac{3M\gamma}{LC^3}q^2\dot{q} + \frac{q}{LC} = 0\,. \tag{4.8}$$

Сделаем замену параметров:

$$2\delta = \frac{M\beta}{LC} - \frac{R}{L};\ a = \left(\frac{M\beta}{LC} - \frac{R}{L}\right)^{-1}\cdot\frac{3M\gamma}{LC^3};\ \omega^2 = \frac{1}{LC}\,.$$

Тогда уравнение (4.8) принимает вид широко известного **уравнения Ван-дер-Поля:**

$$\ddot{q} - 2\delta(1 - aq^2)\dot{q} + \omega^2 q = 0\,. \tag{4.9}$$

В уравнении (4.9) предполагается, что $\delta > 0$, $a > 0$ и $\omega^2 > 0$. При достаточно малых q уравнение (4.9) близко к линейному осциллятору с отрицательным трением, поэтому состояние равновесия $(q = 0, \dot{q} = 0)$ является неустойчивым узлом или фокусом. При достаточно больших по модулю q $(q^2 > 1/a)$ множитель перед $\dot{q}$ в уравнении (4.9) становится положительным и мы вправе предположить, что при этом колебания затухают. Таким образом, малые колебания нарастают, а большие – затухают. Следовательно, между ними должно быть устойчивое периодическое движение Γ (рис. 4.10).

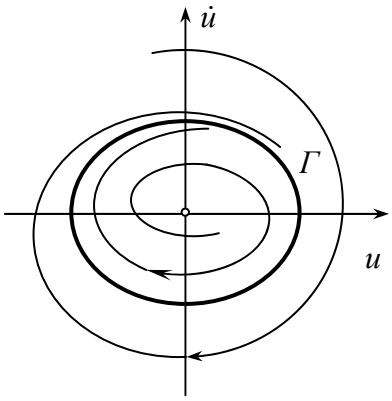

Рис. 4.10

Уравнение Ван-дер-Поля (4.9) можно переписать в виде:

$$\ddot{q} + \omega^2 q = -\varepsilon q^2 \dot{q} + 2\delta \dot{q}\,, \tag{4.10}$$

где $\varepsilon = \dfrac{3M\gamma}{LC^3} > 0$. Если параметры лампового генератора подобраны так, что величины δ и ε – достаточно *малые*, то решение уравнения (4.10) можно искать в виде:

$$q = A\cos(\omega t + \varphi)\,, \tag{4.11}$$

где в силу малости δ и ε величины A и φ являются *медленно меняющимися* функциями времени. Представление (4.11) неоднозначно. Пользуясь этой неоднозначностью, для выделения единственного решения можно добавить к условию (4.11) еще одно условие, например:

$$\dot{q} = -\omega A\sin(\omega t + \varphi)\,, \tag{4.12}$$

после чего A и φ выражаются через q и $\dot{q}$. Далее, можно непосредственно убедиться, что производные $\dot{A}$ и $\dot{\varphi}$, взятые в силу уравнения (4.10), *малы* ввиду малости δ и ε. Это означает, что величины A и φ вдоль решений уравнения (4.10) близки к тождественным (во времени) константам.

Рассмотрим, как меняется со временем энергия осциллятора, отвечающего уравнению (4.10) при $\varepsilon = \delta = 0$. Энергия такого осциллятора равна

$$E = \frac{1}{2}(\dot{q}^2 + \omega^2 q^2),$$

а ее полная производная по времени, согласно (4.10), имеет вид:

$$\frac{dE}{dt} = \dot{q}\ddot{q} + \omega^2 q\dot{q} = (2\delta - \varepsilon q^2)\dot{q}^2$$

или, учитывая (4.11) и (4.12),

$$\frac{dE}{dt} = \left[2\delta - \varepsilon A^2 \cos^2(\omega t + \varphi)\right] A^2\omega^2 \sin^2(\omega t + \varphi). \qquad (4.13)$$

Интегрируя по периоду $2\pi/\omega$ полученное соотношение (4.13) при тех же предположениях $\dot{A} = \dot{\varphi} = 0$, найдем, что изменение энергии за это время равно

$$\Delta E = 2\pi\delta\omega A^2 - \frac{\pi\varepsilon\omega}{2} A^4.$$

С другой стороны, согласно (4.11) и (4.12), $E = \omega^2 A^2/2$. Поэтому

$$\Delta E = \frac{\omega^2}{2}\Delta A^2 = 2\pi\delta\omega A^2 - \frac{\pi\varepsilon\omega}{2} A^4$$

(здесь ΔA^2 – приращение величины A^2 за период), или

$$\Delta A^2 = \frac{4\pi\delta}{\omega} A^2 - \frac{\pi\varepsilon}{\omega} A^4. \qquad (4.14)$$

Обозначая последовательные значения амплитуды A, отстоящие друг от друга на промежутке времени $2\pi/\omega$, через A_n и A_{n+1}, приходим, согласно (4.14), к связывающему их соотношению:

$$A_{n+1}^2 = \left(1 + \frac{4\pi\delta}{\omega}\right) A_{n+1}^2 - \frac{\pi\varepsilon}{\omega} A_n^4. \qquad (4.15)$$

Формула (4.15) задает точечное отображение полупрямой неотрицательных значений A^2 в себя. Это точечное отображение,

как и в случае часов Галилея – Гюйгенса, может быть рассмотрено с помощью диаграммы Ламерея (рис. 4.11, 4.12).

δ < 0, ε > 0

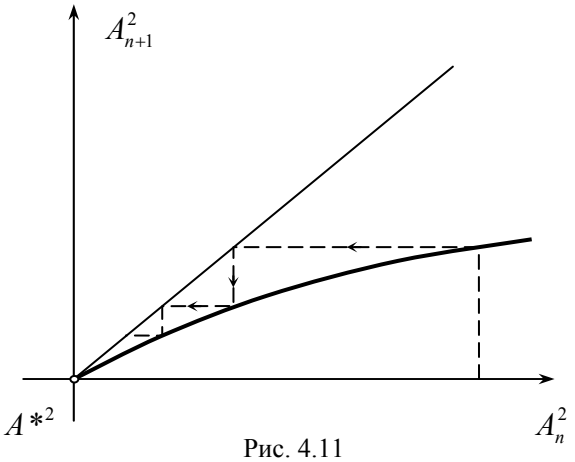

Рис. 4.11

δ > 0, ε > 0

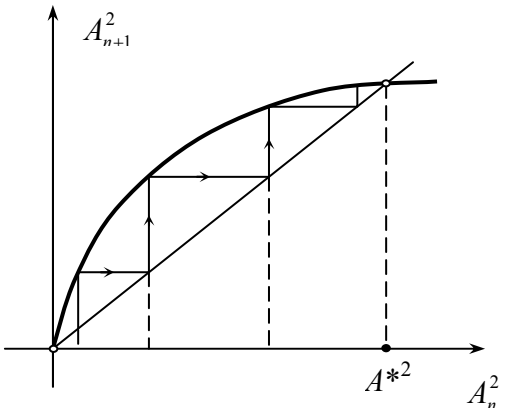

Рис. 4.12

Из диаграммы Ламерея при $\delta < 0$ и постоянном $\varepsilon > 0$, представленной на рис. 4.11, следует, что при $\delta < 0$ амплитуда движения A уменьшается до нуля, т.е. состояние равновесия глобально асимптотически устойчиво.

Если же имеет место интересующий нас случай $\delta > 0$ (рис. 4.12), то из вида диаграммы следует неустойчивость состояния равновесия и наличие глобально устойчивых автоколебаний (периодического движения, предельного цикла). (Автоколебания можно считать *глобально* устойчивыми, так как траектория не приходит в автоколебательный режим только при начальных условиях $q = \dot{q} = 0$, т.е. при *точном* нахождении системы в неустойчивом состоянии равновесия, что практически невозможно из-за наличия флуктуаций.) Амплитуда этих автоколебаний находится как неподвижная точка отображения (4.15) и равна $A^* = 2\sqrt{\delta / \varepsilon}$, а период (при сделанных предположениях малости δ и ε) приближенно равен $2\pi / \omega$.

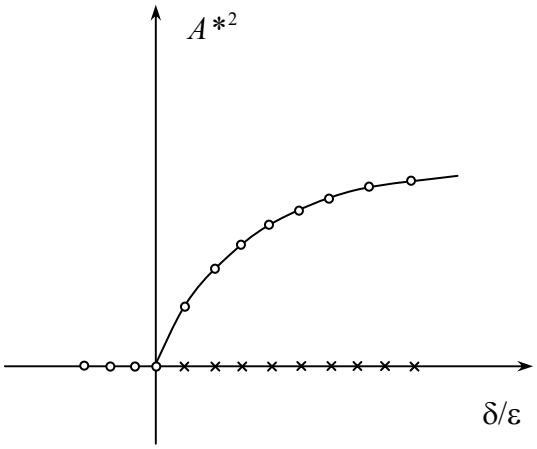

Рис. 4.13

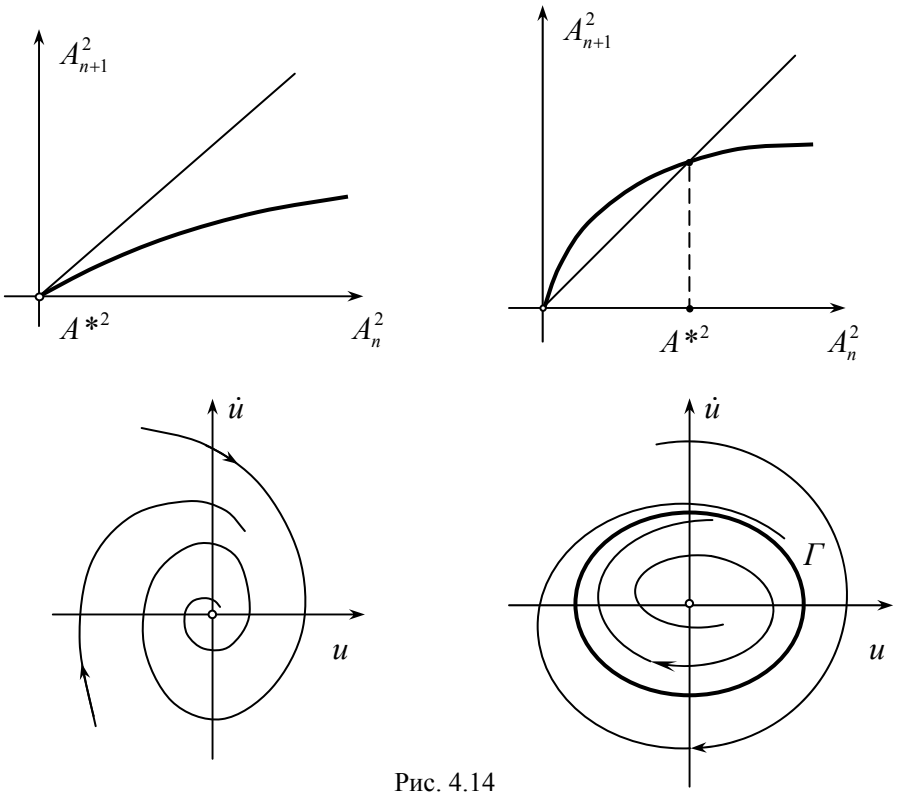

Рис. 4.14

На рис. 4.13 изображена **бифуркационная диаграмма** зависимости амплитуды автоколебаний A^* ($A^* = 0$ соответствует состоянию равновесия) от величины δ / ε (фактически от величины δ, если $\varepsilon = \text{const} > 0$). На этой диаграмме устойчивые режимы движения отмечены кружочками, а неустойчивые – крестиками. При $\delta < 0$ система приходит к глобально устойчивому состоянию равновесия. При смене знака δ состояние равновесия теряет устойчивость и возникают автоколебания, амплитуда которых при росте δ растет *от нуля*, т.е. они как бы рождаются из равновесия, сменяющего устойчивость. При обратном изменении

параметра δ вся картина повторяется в обратном порядке: если система находилась в автоколебательном режиме, то при уменьшении δ амплитуда колебаний уменьшается и при переходе значения δ через нуль автоколебания исчезают, и система приходит к глобально устойчивому состоянию равновесия. Такой режим возбуждения автоколебаний называется **мягким** или **обратимым**. Соответствующие изменения диаграммы Ламерея и фазового портрета показаны на рис. 4.14.

Однако если анодно-сеточную характеристику лампы аппроксимировать по-другому, например полиномом 5-й степени

$$I_a = \alpha + \beta E_g + \gamma E_g{}^3 - \varepsilon E_g{}^5 \quad (\alpha \geq 0\,,\, \beta, \gamma, \varepsilon > 0),$$

то при повторении выкладок для соответствующего дифференциального уравнения мы приходим к точечному отображению вида:

$$A_{n+1}^2 = \left(1 + \frac{4\pi\delta}{\omega}\right) A_n^2 + \varepsilon_1 A_n^4 + \varepsilon_2 A_n^6, \qquad (4.16)$$

где коэффициенты ε_1 и ε_2 пропорциональны соответственно γ и ε и, как и δ, по предположению достаточно малы. При рассмотрении диаграмм Ламерея для отображения (4.16) в случае $\varepsilon_1 < 0$ и $\varepsilon_2 > 0$ в системе можно наблюдать не мягкое, а **жесткое** возбуждение автоколебаний (рис. 4.15).

Вначале (при значениях δ, меньших некоторого отрицательного числа) в системе имеется глобально устойчивая неподвижная точка (состояние равновесия) *O*. Затем при некотором отрицательном значении появляется полуустойчивая неподвижная точка M_{12}, соответствующая полуустойчивому периодическому движению. При дальнейшем увеличении δ это полуустойчивое движение распадается на устойчивое M_2 и неустойчивое M_1 периодические движения, причем амплитуда устойчивого движения увеличивается, а неустойчивого – уменьшается, в результате чего при $\delta = 0$ периодическое движение M_1 сливается с состоянием равновесия *O*, которое из устойчивого становится неустойчивым. Таким образом, при переходе значения δ через нуль тра-

ектория «срывается» с состояния равновесия на периодическое движение *конечной* (достаточно удаленной от нуля) амплитуды. Соответствующие изменения фазового портрета показаны на рис. 4.16.

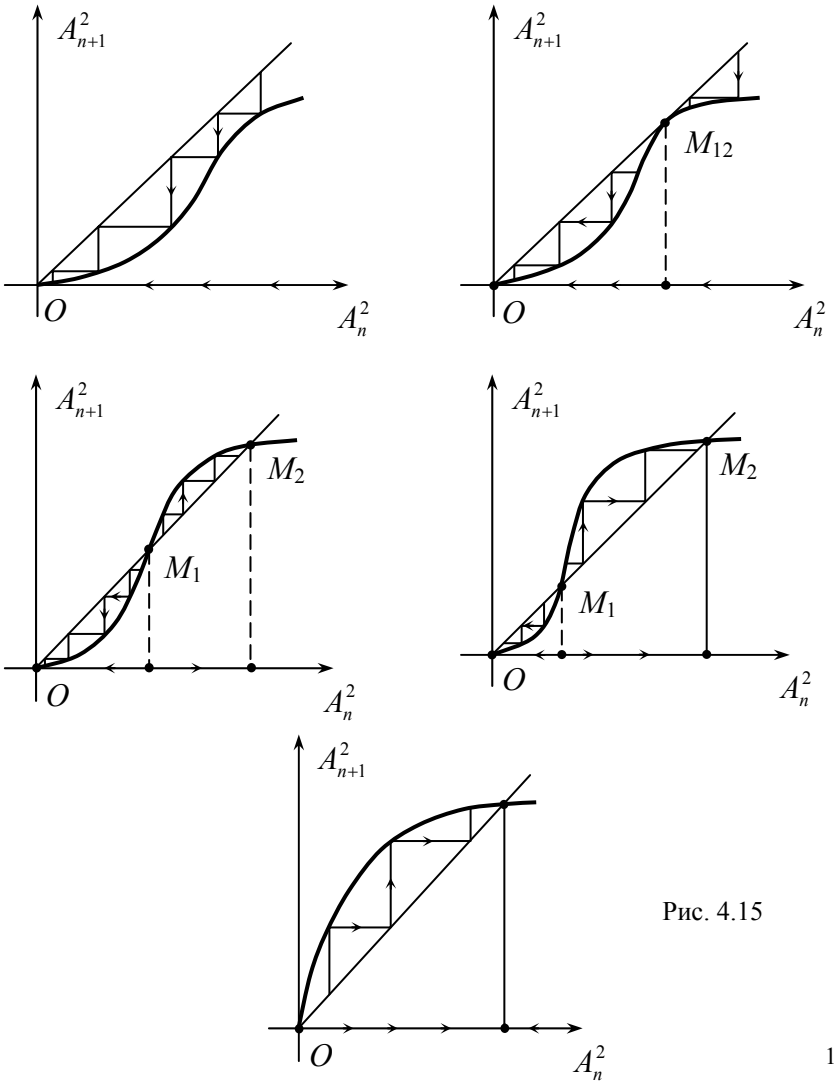

Рис. 4.15

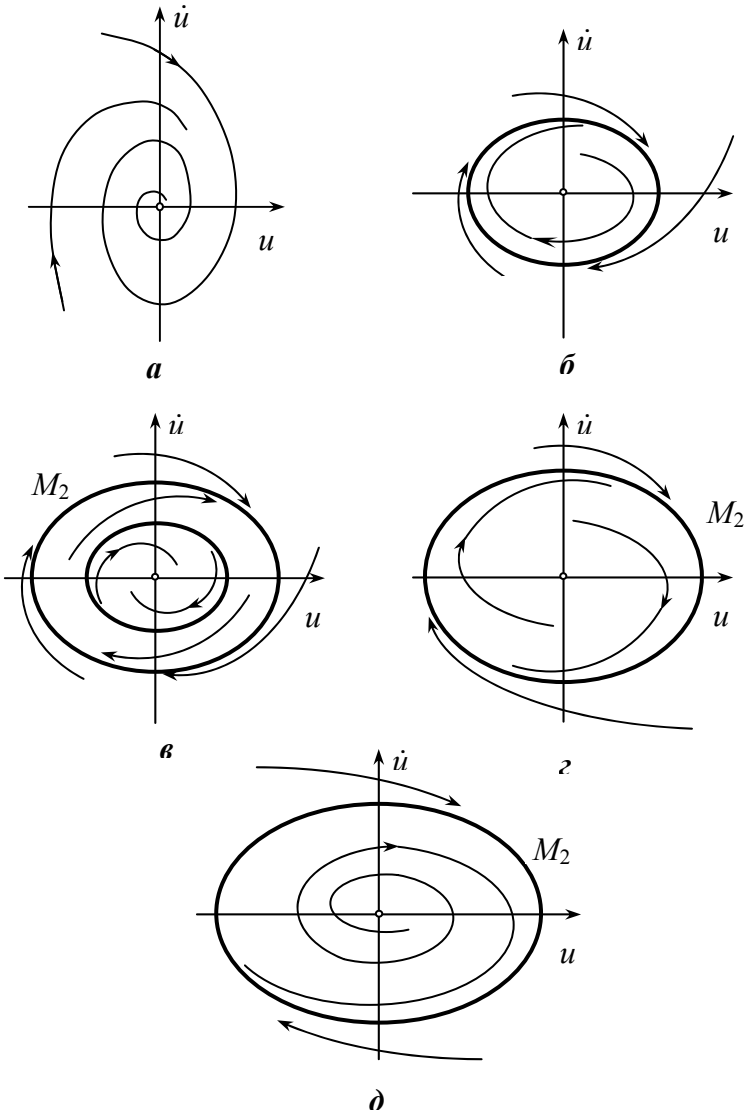

Рис. 4.16

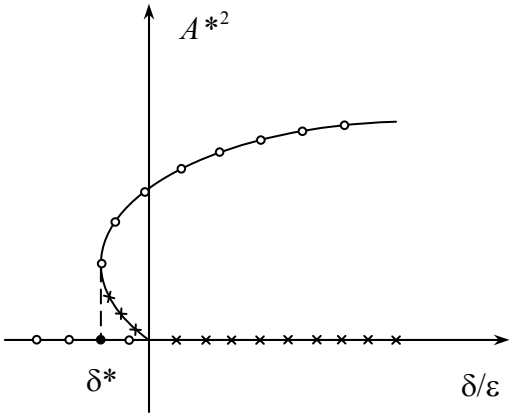

Рис. 4.17

Если система находится в режиме устойчивого периодического движения, то при обратном изменении δ картина *не повторяется*: при переходе δ через нуль система не приходит в устойчивое состояние равновесия *O*, а остается в автоколебательном режиме, амплитуда которого меняется. Поэтому жесткое возбуждение автоколебаний носит **необратимый** характер. Исчезновение автоколебаний наблюдается только при некотором отрицательном значении δ; следовательно, для жесткого возбуждения автоколебаний имеет место **гистерезис** (вспомните вольт-амперную характеристику неоновой лампы!). Бифуркационная диаграмма зависимости амплитуды автоколебаний A^* от δ (точнее, от δ/ε) для жесткого возбуждения автоколебаний показана на рис. 4.17 (кружочками отмечены устойчивые режимы движения, а крестиками – неустойчивые).

4.3.2. Реализация в AnyLogic

Работа реализована в файле `Part3\generator_new.alp` (рис. 4.18). В окне анимации можно наблюдать прохождение анодного тока через лампу, изменение величины и знака (поляр-

ности) заряда конденсатора, а также изменение магнитного потока, создаваемого током, протекающим через катушку (яркость силовых линий магнитного поля катушки). При этом изменению величин тока и заряда соответствует изменение яркости пластин конденсатора и стрелки справа от лампы (направление стрелки всегда соответствует направлению тока – от анода к катоду).

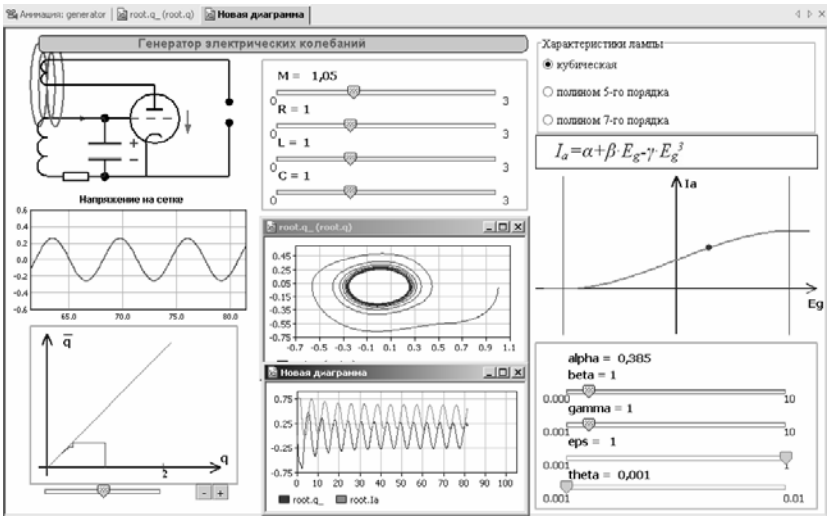

Рис. 4.18

Внутри анимации выводятся осциллограмма изменения напряжения на сетке лампы и диаграмма Ламерея, а также движение точки по графику анодно-сеточной характеристики лампы. Диаграмма Ламерея здесь строится не для приближенного отображения амплитуды в себя, а для отображения заряда q в себя при условии $\dot{q}=0$. С помощью кнопок «–» и «+», расположенных справа снизу от диаграммы Ламерея, можно отмасштабировать диаграмму.

Фазовый портрет и осциллограммы анодного тока и тока через контур выводятся в отдельных окнах (чтобы увидеть их, нуж-

но в окне анимации выбрать вкладки «Новая диаграмма» и «root.q_(root.q)»).

Аппроксимация характеристики лампы (кубическая, полином 5-й или 7-й степени) выбирается с помощью радиокнопки. С помощью бегунков можно изменять параметры контура и коэффициенты полиномов, аппроксимирующих характеристику лампы.

4.3.3. Задания для эксперимента

1. Выбирая с помощью радиокнопки аппроксимацию характеристики лампы, проследите за бифуркациями состояния равновесия (потеря устойчивости, рождение предельных циклов) для каждой аппроксимации. Сколько существует устойчивых периодических движений в случае аппроксимации характеристики лампы полиномом 3-й степени? 5-й степени? 7-й степени? Мягко или жестко возбуждаются автоколебания? Как это отражается на диаграмме Ламерея?

2. Как меняется период автоколебаний при изменении параметров системы? Как он зависит от аппроксимации характеристики лампы?

4.4. Динамика вертикального полета мячика с отбором энергии в верхней точке

4.4.1. Модель

Рассмотрим процесс полета мячика массы m в поле силы тяжести вверх и вниз (для простоты трением о воздух пренебрежем). Полет мячика будет описываться вертикальной координатой y – расстоянием от мячика до пола. Будем считать, что при ударе о пол мячик отскакивает от него с коэффициентом отскока (по скорости) k, который может быть больше единицы, меньше единицы (но больше нуля) или равен единице:

$$\dot{y}(t_{\text{удара}} + 0) = -k\dot{y}(t_{\text{удара}} - 0),$$

где $t_{\text{удара}}$ – момент удара мячика о пол.

Пусть при достижении некоторой заданной постоянной высоты H (при полете мячика вверх, т.е. при $\dot{y} > 0$) у мячика отбирается часть энергии, равная постоянной величине $\Delta E > 0$ (если мячик не долетает до высоты H, то энергия не отбирается). При этом вначале будет отбираться часть кинетической энергии, а если кинетическая энергия меньше ΔE – то вся кинетическая и часть потенциальной энергии, что приведет к резкому уменьшению высоты мячика над полом. Если же полная механическая энергия меньше ΔE, то у мячика отнимается вся его энергия, и он скачком приходит в состояние равновесия $(y = 0, \dot{y} = 0)$.

Пусть S – полная механическая энергия мячика после очередного удара о пол, а $\overline{S}$ – полная механическая энергия мячика после *следующего* удара о пол. Тогда математическая модель движения мячика (в виде точечного отображения – функции последования) имеет вид:

$$\overline{S} = \begin{cases} k^2 S \quad (S \le mgH); \\ k^2 (S - \Delta E) \quad (S > mgH \text{ и } S \ge \Delta E); \\ 0 \quad (mgH < S < \Delta E). \end{cases} \tag{4.17}$$

В зависимости от k, а также от соотношения между mgH, ΔE и текущим значением S у отображения (4.17) может быть пять видов диаграмм Ламерея:

1. При $0 < k < 1$ и $mgH \ge \Delta E$ (рис. 4.19). В этом случае точечное отображение имеет единственную глобально устойчивую неподвижную точку $S = 0$, которая соответствует глобально асимптотически устойчивому *состоянию равновесия*, т.е. полной остановке движения мячика.

2. При $0 < k < 1$ и $mgH < \Delta E$ (рис. 4.20). В этом случае при любых начальных условиях мячик также останавливается, но в отличие от предыдущего случая полное затухание колебаний мя-

чика может произойти за конечное время, если точка придет на отрезок $S_1 \le S \le S_2$, переходящий за один шаг отображения (4.17) в неподвижную точку $S = 0$. Применяя отображение, обратное к (4.17), можно убедиться, что у отрезка $S_1 \le S \le S_2$ имеется счетное множество прообразов, причем чем правее на оси S расположен отрезок-прообраз, тем больше его длина.

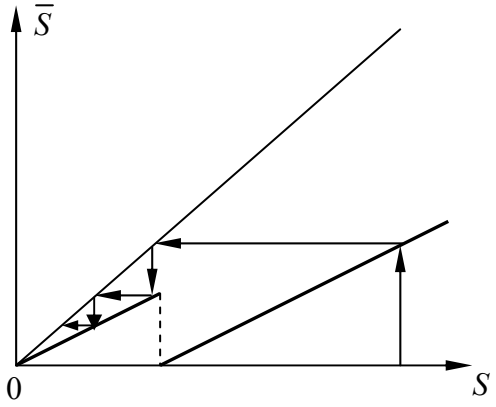

Рис. 4.19

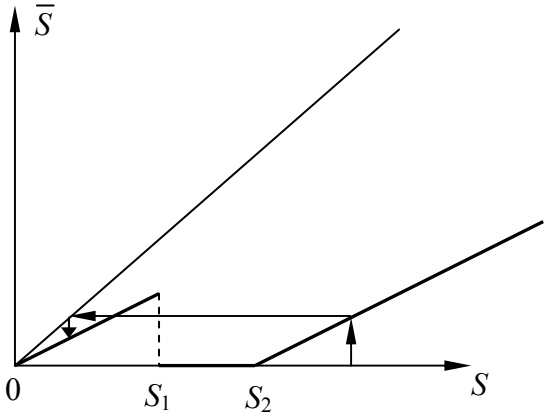

Рис. 4.20

3. При $k > 1$ и небольших значениях ΔE (ΔE меньше некоторого критического значения ΔE_1) (рис. 4.21). В этом случае у отображения (4.17) имеется единственная неподвижная точка $S = 0,$ являющаяся неустойчивой и соответствующая неустойчивому состоянию равновесия. При любых начальных условиях диаграмма Ламерея уходит в бесконечность, т.е. мячик с каждым новым отскоком от пола будет улетать всё дальше вверх.

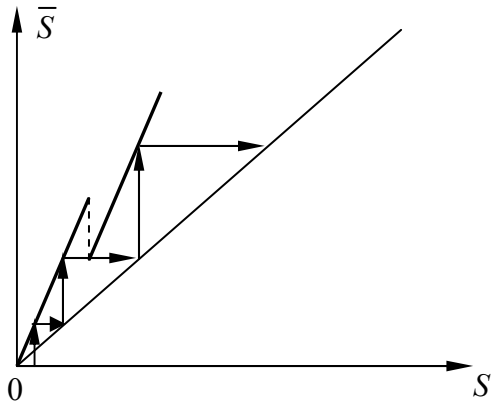

Рис. 4.21

4. При $k > 1$ и значениях ΔE, больших ΔE_1, но меньших некоторого другого критического значения ΔE_2 (рис. 4.22). В этом случае отображение (4.17) имеет две неподвижные точки: $S = 0,$ соответствующая неустойчивому *состоянию равновесия*, и $S = S^* > 0,$ соответствующая неустойчивому *периодическому движению* на фазовой плоскости $(y, \dot{y})$. При любых начальных условиях диаграмма Ламерея уходит в бесконечность (если начальное S меньше, чем S^*, то последовательные значения S некоторое время будут находиться в области $S < S^*$, но на некотором шаге значение S обязательно станет больше S^*, и диаграмма Ламерея уйдет в бесконечность).

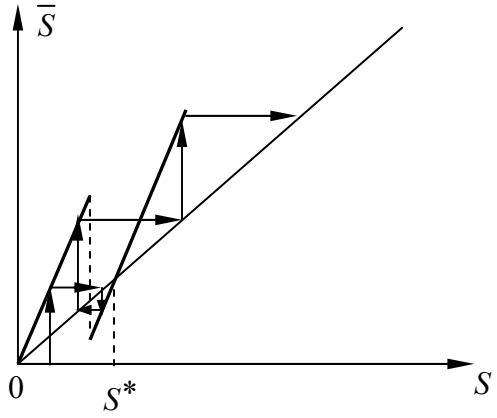

Рис. 4.22

5. При $k > 1$ и значениях ΔE, больших ΔE_2 (рис. 4.23).

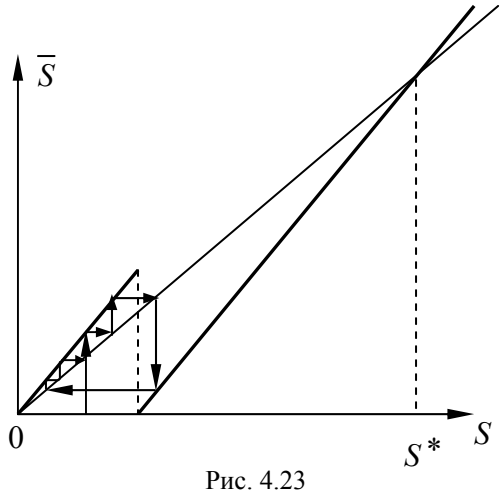

Рис. 4.23

Как и в случае 4, у отображения (4.17) имеются две неустойчивые неподвижные точки: $S = 0$, соответствующая неустойчи-

вому *состоянию равновесия*, и $S = S^* > 0$, соответствующая неустойчивому *периодическому движению* на фазовой плоскости $(y, \dot{y})$. Поведение фазовых траекторий зависит от начальных условий. Если начальное значение координаты или скорости мячика достаточно велико, так что $S > S^*$, то диаграмма Ламерея уходит в бесконечность, и мячик с каждым новым отскоком от пола будет улетать всё дальше вверх.

Если же начальное значение полной механической энергии мячика не очень велико ($S < S^*$), то диаграмма Ламерея обязательно придет на некоторый отрезок $0 \le S \le S_{\max}$ ($S_{\max} < S^*$) и далее будет всё время находиться на этом отрезке, ни разу не возвращаясь в точки, в которых диаграмма (траектория) уже побывала, и происходит сплошное заполнение этого отрезка последовательными значениями *S*. При этом можно показать, что сколь бы близко не находились две начальные точки S_0 и $\widetilde{S}_0$ на диаграмме Ламерея, в некоторые моменты времени траектории отображения (4.17), выпущенные из этих точек, обязательно разойдутся на расстояние, большее, чем $\Delta E / 2$ (при этом траектории остаются ограниченными). Это означает, что если мы *измеряем* начальное значение *S* (а измерение обязательно происходит с некоторой *погрешностью*), то при прогнозировании изменения *S* с помощью отображения (4.17) мы получим через несколько шагов *совершенно другой результат*, чем тот, который должен был бы иметь место при *точном* знании начальных условий. Таким образом, поведение динамической системы в этом случае *практически непредсказуемо*. Такая структура диаграммы Ламерея или фазового портрета называется **(динамическим) хаосом** или **хаотическим аттрактором** (слово «аттрактор» означает «притягиватель»; это значит, что при любых начальных условиях, если только $S < S^*$, траектория обязательно войдет в область хаоса и навсегда останется там).

Можно показать, что хаотическое поведение траекторий допускает описание на языке теории вероятностей (можно найти плотность вероятности попадания точки на диаграмме Ламерея в

подынтервалы интервала $0 < S < S^*$), т.е. хаотические движения не только практически непредсказуемы, но и *случайны*. При этом плотность вероятности при стремлении числа шагов отображения (4.17) к бесконечности будет стремиться к *равномерному распределению на отрезке* $0 \le S \le S_{\max}$. Это означает, что вероятность попадания значения S в подынтервалы отрезка $0 \le S \le S_{\max}$, имеющие одинаковую длину, одинакова.

4.4.2. Реализация в AnyLogic

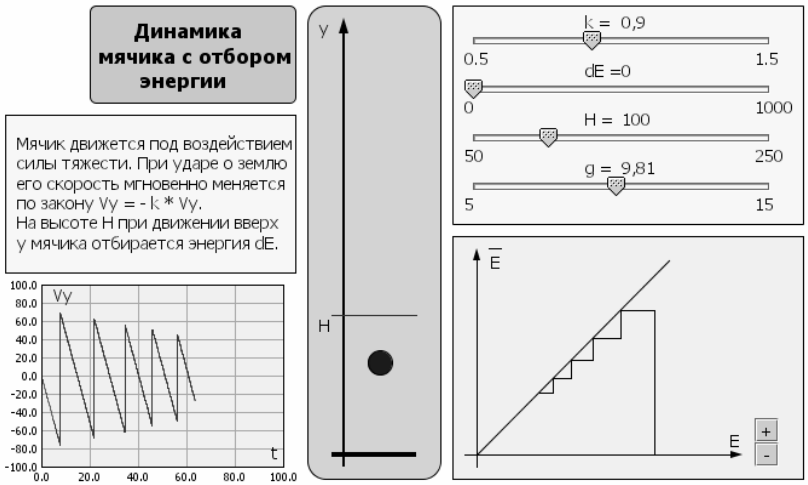

Рис. 4.24

Работа реализована в файле `Part3\ball.alp` (рис. 4.24). В окне анимации можно наблюдать летающий мячик, а также осциллограмму изменения скорости (обозначенной в анимации через `Vy`) и диаграмму Ламерея для отображения полной механической энергии (она обозначена здесь буквой *E*) в себя. С помощью бегунков можно менять параметры системы. В качестве началь-

ных условий здесь взяты значения $y = 300$ и $\dot{y} = 0$. Параметр ΔE обозначен в анимации как dE.

4.4.3. Задания для эксперимента

Изменяя значения параметров системы, добейтесь входа мячика поочередно в каждый из возможных режимов движения (затухание колебаний, уход в бесконечность, хаос). Как влияют значения параметров k, ΔE и H на режим движения мячика?

5. Имитационное (агентное) моделирование динамических систем

5.1. Имитационная модель «хищник – жертва» с учетом насыщения

5.1.1. Модель

Имитационная модель «хищник – жертва» – это модель сосуществования двух популяций, одна из которых (хищники, например рыси) поедает другую (жертвы, например зайцы). В данной модели считается, что жертвы могут гибнуть либо по истечении срока продолжительности жизни, либо от поедания хищниками, либо в результате внутривидовой конкуренции (борьбы жертв за пищу и среду обитания). Хищники же в рамках модели могут погибать либо по истечении срока продолжительности жизни, либо от голода (вследствие нехватки жертв).

Пространство, в котором обитают хищники и жертвы, разделено на клетки, в каждой из которых может быть не более определенного количества хищников и жертв.

В базовой модели учитываются следующие параметры.

1. Для жертв:
   - интенсивность рождения;
   - число детенышей, рождающихся за один раз;
   - продолжительность жизни одной особи;
   - коэффициент внутривидовой конкуренции.
2. Для хищников:
   - интенсивность рождения;
   - число детенышей, рождающихся за один раз;
   - продолжительность жизни одной особи;
   - период, в течение которого хищник охотится на жертву (в днях);
   - время, через которое хищник погибает от голода при отсутствии жертвы (в днях).

Отметим, что базовая модель поставляется вместе с системой

AnyLogic в виде файла `PredatorPrey.alp`. В настоящей модели добавлена возможность учета насыщения хищников. За насыщение отвечают следующие параметры:

- максимальное количество жертв в одной клетке;
- интенсивность поедания жертв;
- коэффициент насыщения хищника.

Учитывается также возможность *случайных миграций* хищников и жертв из клетки в клетку.

Если рассматривать сосуществование видов в непрерывном времени, считая численности популяций *непрерывно изменяющимися*, то процесс взаимодействия хищников и жертв можно приближенно описать системой **дифференциальных уравнений типа Вольтерра – Лотки**:

$$\left.\begin{aligned} \dot{x} &= a_1 x - \frac{b_1 x y}{1 + Ax} - c_1 x^2, \\ \dot{y} &= -a_2 y + \frac{b_2 x y}{1 + Ax}, \end{aligned}\right\} \qquad (5.1)$$

где:

x – численность популяции жертвы;

y – численность популяции хищника;

a_1 – коэффициент размножения жертвы (считается, что при отсутствии хищника жертва размножается экспоненциально);

a_2 – коэффициент вымирания хищника (считается, что при отсутствии жертвы хищник вымирает экспоненциально);

b_1 – коэффициент уменьшения численности жертвы за счет поедания ее хищником;

b_2 – коэффициент увеличения численности хищника за счет поедания жертвы;

A – коэффициент насыщения хищника (чем больше жертвы успел съесть хищник, тем медленнее он будет поедать жертву в дальнейшем);

c_1 – коэффициент внутривидовой конкуренции жертвы.

Все фазовые переменные системы (5.1) неотрицательны, а

параметры положительны.

5.1.2. Реализация в AnyLogic

Работа реализована в файле `Part4\PredatorPrey.alp` (рис. 5.1).

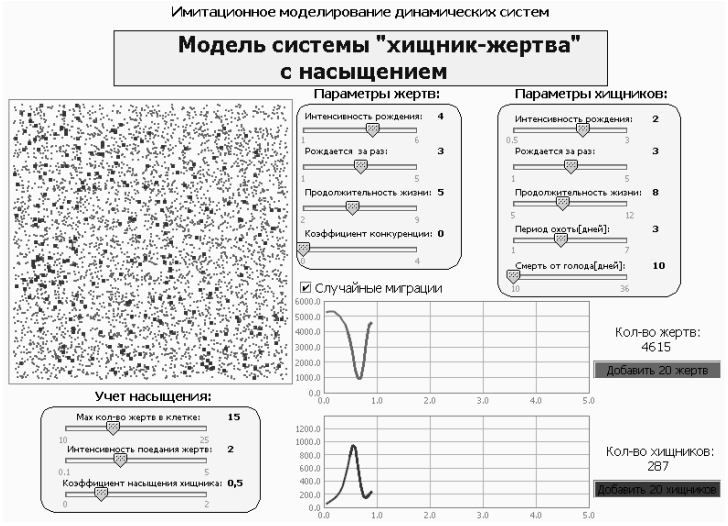

Рис. 5.1

В окне анимации можно наблюдать поле, по которому перемещаются особи популяций хищника и жертвы (показаны двумя разными цветами), при этом их численность изменяется. С помощью бегунков можно изменять параметры системы, а с помощью флажка – включать и отключать возможность случайных миграций особей. Динамика изменения численностей популяций видна на соответствующих диаграммах (осциллограммах), при этом время измеряется в годах. Две дополнительные кнопки в правой части окна позволяют «вбросить» в систему дополнительно 20 жертв или 20 хищников. В начальный момент времени в системе имеется 5000 особей жертвы и 40 особей хищника.

5.1.3. Задания для эксперимента

1. Пронаблюдайте, как влияет изменение параметров имитационной модели на динамику численностей популяций (в частности, на амплитуду и период колебаний). Возможна ли ситуация полного вымирания какой-либо популяции?

2. Добавьте некоторое количество жертв или хищников в систему. Как это повлияет на динамику численности популяций?

3*. Реализуйте в AnyLogic модель типа Вольтерра – Лотки (5.1) с выводом осциллограмм на экран. Как подобрать параметры модели (5.1) так, чтобы динамика модели (5.1) была максимально близка к динамике имитационной модели?

5.2. Модель химической кинетики – брюсселятор

5.2.1. Модель

В настоящее время известно достаточно много *периодических* химических реакций, т.е. реакций, в процессе протекания которых наблюдаются автоколебания количеств (или концентраций) различных веществ. Для лучшего понимания таких реакций разработаны некоторые простые математические модели. Одна из таких моделей была предложена и исследована в 1968 г. в Брюсселе коллективом ученых во главе с И. Пригожиным, в связи с чем эта модель получила название **«брюсселятор»**. Реакция, описываемая моделью брюсселятора, протекает в **открытой системе** (исходные реагенты могут подводиться в систему извне, а конечные продукты реакции выводятся из системы).

Как уже отмечалось в п. 1.3.3, *химические* уравнения брюсселятора имеют вид:

$$A \rightarrow X,\ B+X \rightarrow Y+D,\ 2X+Y \rightarrow 3X,\ X \rightarrow E. \qquad (5.2)$$

Здесь A и B – исходные реагенты, D и E – конечные продукты реакции, X и Y – промежуточные продукты реакции.

Обозначая концентрации реагирующих веществ в (5.2) теми же буквами и считая A и B заданными константами, получим *дифференциальные* уравнения брюсселятора [3]:

$$\left.\begin{aligned} \dot{X} &= A-(B+1)X+X^2Y, \\ \dot{Y} &= BX-X^2Y. \end{aligned}\right\} \qquad (5.3)$$

5.2.2. Реализация в AnyLogic

Работа реализована в файле `Part4\Brussel2D.alp` (рис. 5.2).

Рис. 5.2

Анимация для имитационной модели брюсселятора аналогична анимации для имитационной модели сосуществования видов (п. 5.1.2). Можно наблюдать динамику количеств молекул веществ A, B, X и Y (точки четырех разных цветов), осциллограммы их изменения во времени, изменять параметры распада и слияния молекул (средние времена), добавлять новые молекулы.

В начальный момент времени в системе имеется 800 молекул вещества A, 50 молекул вещества B и ни одной молекулы веществ X и Y.

Технология создания данной имитационной модели подробно описана в п. 1.3.3.

5.2.3. Задания для эксперимента

1. Пронаблюдайте, как влияет изменение параметров имитационной модели на динамику количеств (концентраций) веществ (в частности, на амплитуду и период колебаний).

2. Добавьте некоторое количество молекул вещества *X* или *Y* в систему. Как это повлияет на динамику количеств (концентраций) веществ?

3*. Реализуйте в AnyLogic дифференциальную модель брюсселятора (5.3) с выводом осциллограмм на экран. Как подобрать параметры модели (5.3) так, чтобы динамика модели (5.3) была максимально близка к динамике имитационной модели?

4*. Реализуйте в AnyLogic имитационную модель химической реакции, предложенной в [5]:

$$A \rightarrow X,\ B+X \rightarrow Y+D,\ 2X+Y \rightarrow 3X,\ X \rightarrow Z,\ Z+Y \rightarrow X, \qquad (5.4)$$

где Z – дополнительное вещество, играющее роль модулятора концентрации вещества X. Эта модуляция может привести к **хаосу**, т.е. к такому типу движений, при котором траектории, начинающиеся из произвольных сколь угодно близких начальных точек, за небольшое время обязательно разойдутся на расстояние, большее некоторого одного и того же числа [3] (пример системы с хаотической динамикой рассматривался в п. 4.4). Можно ли подобрать параметры имитационной модели так, что концентрации веществ X, Y и Z будут изменяться хаотически?

5*. Попробуйте составить дифференциальную модель, соответствующую химическим уравнениям (5.4). Удастся ли при изменении ее параметров наблюдать автоколебания и хаотические движения?

Список литературы

1. Неймарк Ю.И. Математические модели в естествознании и технике. Н.Новгород: Изд-во ННГУ, 2004.
2. Неймарк Ю.И., Коган Н.Я., Савельев В.П. Динамические модели теории управления. М.: Наука, Главная редакция физ.-мат. литературы, 1985.
3. Неймарк Ю.И., Ланда П.С. Стохастические и хаотические колебания. М.: Наука, Главная редакция физ.-мат. литературы, 1987.
4. Рабинович М.И., Трубецков Д.И. Введение в теорию колебаний и волн. М.: Наука, Главная редакция физ.-мат. литературы, 1984.
5. Ланда П.С. Автоколебания в распределенных системах. М.: Наука, Главная редакция физ.-мат. литературы, 1983.