НАУЧНАЯ БИБЛИОТЕКА

МАИ НАУЧНАЯ БИБЛИОТЕКА

А.Ю. Морозов, Д.Л. Ревизников

МЕТОДЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ ДИНАМИЧЕСКИХ СИСТЕМ С ИНТЕРВАЛЬНЫМИ ПАРАМЕТРАМИ

А.Ю. Морозов, Д.Л. Ревизников

Издательство МАИ

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Московский авиационный институт
(национальный исследовательский университет)

А.Ю. МОРОЗОВ
Д.Л. РЕВИЗНИКОВ

МЕТОДЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ ДИНАМИЧЕСКИХ СИСТЕМ С ИНТЕРВАЛЬНЫМИ ПАРАМЕТРАМИ

Москва
Издательство МАИ
2019

ББК 22.19
УДК 519.622.2:519.652
М80

Морозов А.Ю., Ревизников Д.Л. Методы компьютерного моделирования динамических систем с интервальными параметрами. — М.: Изд-во МАИ, 2019. — 160 с.: ил.

Монография посвящена созданию эффективных методов моделирования динамических систем с интервальными параметрами. Приведен обзор известных методов и их программных реализаций. Представлен алгоритм адаптивной интерполяции на основе kd-дерева, который позволяет находить интервальные оценки решений с контролируемой точностью. Суть алгоритма заключается в построении динамической структурированной сетки на основе kd-дерева над множеством, образованным интервальными параметрами задачи. При этом оценка глобальной погрешности метода прямо пропорциональна высоте kd-дерева. Тестирование алгоритма на ряде представительных задач разной размерности и содержащих разное количество интервальных параметров, демонстрирует его эффективность.

Монография предназначена для широкого круга научных работников, специализирующихся на решении задач с интервальными неопределенностями. Она может быть полезна преподавателям, аспирантам и студентам старших курсов.

Рецензенты:
д-р физ.-мат. наук *С.И. Мартыненко*;
д-р физ.-мат. наук, профессор *С.И. Мухин*

ISBN 978-5-4316-0651-9

ВВЕДЕНИЕ

При решении прикладных задач механики, химической кинетики, газовой динамики и других или при исследовании определенных свойств динамических систем часто возникают ситуации, когда какие-либо параметры точно не известны, но есть информация о диапазонах, в которых находятся их значения. Для таких задач является актуальным получение интервальных оценок решений по известным интервальным значениям их параметров. Традиционно подобные задачи формулируются в виде задачи Коши для системы обыкновенных дифференциальных уравнений (ОДУ) с интервальными начальными условиями или параметрами.

Выделим несколько групп существующих методов.

К первой группе отнесем методы типа Монте-Карло, представленные в работах И.М. Соболя [1], С.М. Ермакова, Г.А. Михайлов [2], Г. Крамера [3], В.М. Золотарева [4] и др. Их суть заключается в проведении многократных расчетов со случайными значениями соответствующих интервальных параметров. Этим методам присущ ряд положительных свойств, таких, как простота, высокая степень распараллеливания и отсутствие потребности в аналитической записи правой части ОДУ, но при этом они обладают низкой ($\sqrt{N}$, где N — количество симуляций) скоростью сходимости и требовательны к вычислительным ресурсам.

В качестве альтернативы выступают методы, в основе которых лежат интервальные вычисления. Еще Архимед использовал интервальные оценки для определения числа π, но активное развитие интервальные методы получили начиная с XX века, в работах Р. Янга [5], П. Двайера [6], М. Вармуса [7], Т. Сунаги [8], Р. Мура [9], Р. Лонера [10], Э. Хансена [11], Г. Алефельда [12], Р. Кравчика [13], К. Никельи [14], А. Ноймайера [15] и др. Среди отечественных родоначальников интервальной математики — В.М. Брадис [16], Л.В. Канторович [17], Ю.И. Шокин [18, 19], Б.С. Добронец [20, 21], С.П. Шарый [22], А.Н. Рогалев [23–25] и др. Из-за своей природы интервальные методы подвержены так называемому эффекту обертывания (эффекту Мура), проявляющемуся в безграничном росте ширины получаемых интервальных оценок решений. Этот эффект

возникает вследствие замены точной формы множества решений на более простую форму и для итеративных методов зачастую приводит к экспоненциальному расхождению границ интервалов. Многие из существующих и разрабатываемых методов направлены на борьбу с этим эффектом [26].

Во вторую группу входят методы, основанные на рядах Тейлора. Они в первую очередь ориентированы на получение гарантированных оценок решений, однако эффект обертывания уменьшают не в полной мере. Их идея заключается в аналитическом разложении решения системы ОДУ в ряд Тейлора с последующей оценкой остаточного члена. Для снижения эффекта Мура выполняется запоминание линейных преобразований, которые производились над множеством решений в процессе вычислений. К этим методам относятся метод Мура [9], метод параллелепипедов [27], QR-метод Лонера [10] и др. Они просты и нетребовательны в плане вычислительных ресурсов, но эффективны в задачах, где интервалы не слишком велики или где нелинейность системы ОДУ проявляется слабо. Существует несколько библиотек, реализующих эти методы: AWA [28], VNODE [29], ADIODES [30], VSNODE [31], VNODE-LP [32].

Библиотека AWA разработана на Pascal-XSC в 1994 году Р. Лонером для решения ОДУ с гарантированной оценкой погрешности.

Библиотека VNODE-LP — это решатель интервальных систем ОДУ, разработанный в 2006 году Н.С. Недялковым на алгоритмическом языке C++. Она является преемником библиотеки VNODE, разработанной этим же автором в 2001 году. Отличительная особенность данного решателя — то, что он полностью посвящен «грамотному программированию» (концепция, введенная Д. Кнутом в 1984 году).

Библиотека VSNODE разработана на языке C++ двумя авторами — Ю. Лин и М. Штадтером в 2005 году. VSNODE является своеобразным расширением библиотеки VNODE для более эффективного решения задач Коши с интервальными параметрами.

Отдельно стоят методы модели Тейлора, разработанные в Мичиганском университете М. Берзом, К. Макино [33–38], М. Нехером, К.Р. Джексоном, Н.С. Недялковым [39], П.С. Натарым, С. Сонду-

ром [40] и др. В их основе лежит принципиально иной подход. Как и в других методах, основанных на рядах Тейлора, решение здесь раскладывается в ряд Тейлора и выполняется оценка остаточного члена, но при этом все вычисления производятся не в числах и не в интервалах, а в так называемых моделях Тейлора. Модель Тейлора представляет собой полином некоторой степени с интервальным остатком. Она сочетает интервальную арифметику с символьными вычислениями. Эффект обертывания уменьшается за счет установки функциональных зависимостей между начальными условиями и решением в каждый момент времени. Все вычисления с коэффициентами полиномиальной части модели Тейлора выполняются на множестве вещественных чисел, а интервальные границы вычисляются только для остатка ряда. При данном подходе все арифметические операции и стандартные функции должны работать с целыми полиномами в качестве операндов. Такой подход довольно значительно снижает эффект обертывания, но при этом работа с полиномами выполняется намного медленнее работы с интервалами [41]. Среди библиотек, реализующих данные методы, можно выделить несколько: COSY Infinity [42], RiOT [43], FlowStar [44, 45], verifyode [46] и др. Область сходимости рядов практически всегда ограничена, поэтому в общем случае для решения задач, содержащих «большие» интервалы, в работах К. Макина, М. Берза [47], М. Клеттинга, А. Рауха, Х. Ашеманна, Е.П. Хофера [48] и др. описываются идеи расщепления исходной области неопределенности на подобласти, которые обрабатываются по отдельности.

Библиотека COSY Infinity разрабатывается в Мичиганском университете профессором Берзом и его командой (последняя версия — 2018 года). Это система для проведения различных современных научных вычислений. Она состоит из следующих частей: набор расширенных и оптимизированных типов данных; объектно ориентированный скриптовый язык COSYScript, который поддерживает полиморфизм и имеет компактный и простой синтаксис; интерфейсы для языков C, F77, C++ и F90 для использования типов данных во внешних программах; различные прикладные пакеты с использованием типов данных COSY, включая физику луча.

Библиотека RiOT — это свободная программа на языке C++ для интегрирования систем ОДУ с интервальными начальными условиями, разработанная в 2008 году в рамках диссертационного исследования в Университете Карлсруэ. В настоящее время доказано, что она может давать недостоверные результаты [49].

Библиотека FlowStar — это инструмент для достоверного моделирования гибридных динамических систем с интервальными параметрами, разработанный на языке C++ (последняя версия — 2017 года). Библиотека состоит в основном из двух модулей: вычислительной библиотеки и алгоритмов высокого уровня. В первый модуль входят реализации интервалов и интервальных арифметик, а также модель Тейлора. Во втором модуле реализованы методы, которые используются в анализе достижимости.

К третьей группе отнесем методы, основанные на символьных вычислениях. В частности, это методы модели Тейлора, а также метод, аппроксимирующий оператор сдвига вдоль траектории (А.Н. Рогалев [23–25]). Как и методы модели Тейлора, метод сдвига вдоль траектории в каждый момент времени получает решение в виде символьного выражения относительно интервальных параметров. Эти методы не подвержены или слабо подвержены эффекту обертывания, справляются с широким классом задач, но при этом для них характерна высокая вычислительная сложность и трудности при распараллеливании.

В работах Б.С. Добронца [50, 51], С.А. Некрасова [52] и др. приводятся методы, способные находить оптимальные двусторонние оценки решений для систем ОДУ, обладающих определенными свойствами. В основном они построены на теоремах сравнения и, как следствие, применимы только для определенных классов систем ОДУ.

Также стоит упомянуть методы, которые приближают множество решений эллипсоидами, параллелепипедами или многогранниками [53, 54]. Они не лишены свойства завышения оценок, и для них желательной является выпуклость множества.

В общем случае для подавления эффекта обертывания необходимо устанавливать зависимость между интервальными параметрами и решением в каждый момент времени. Зачастую сложность соответствующих методов является экспоненциальной относительно количества

интервальных начальных условий или параметров, поэтому очень важно, чтобы они хорошо распараллеливались. В последние годы активно идет развитие программно-аппаратной технологии CUDA, которая позволяет использовать графические процессоры (GPU) компании NVIDIA для общих вычислений. Ключевое отличие GPU от центральных процессоров (CPU) заключается в наличии тысяч ядер, способных одновременно производить расчеты. При использовании даже не очень дорогих видеокарт можно получить прирост производительности в десятки, а то и в сотни раз по сравнению с вычислениями на центральном процессоре. Отметим, что при всей своей привлекательности разработка, ориентированная на GPU, обладает рядом особенностей. Например, вместо наличия только оперативной памяти здесь имеется шесть различных ее видов и возможность напрямую взаимодействовать с кеш-памятью.

Исторически интервальные методы возникли в связи с потребностью в гарантированных вычислениях, которые учитывали бы погрешность самих вычислительных схем, а также ошибки округления при расчетах на ЭВМ. В настоящее время интерес представляют задачи, в которых интервальность возникает непосредственно в самой постановке, а свойством гарантированности можно пренебречь, если есть возможность контролировать точность получаемых границ решений. В связи с этим существует потребность в методах, которые позволяют за приемлемое время находить интервальную оценку решений с контролируемой точностью, не подвержены эффекту обертывания, имеют высокую степень распараллеливания, справляются с «большими» интервалами и при этом не требуют аналитической записи правой части ОДУ и вычисления старших производных.

Суть рассматриваемого в настоящей монографии подхода [55–62] заключается в построении для каждого момента времени кусочно-полиномиальной функции, интерполирующей зависимость решения задачи от точечных значений интервальных параметров с заданной точностью. Для построения такой функции над множеством, образованным интервальными начальными условиями и параметрами задачи, строится адаптивная иерархическая сетка (адаптивное разбиение пространства, AMR [63]), каждая ячейка которой содержит в себе интерполяционную сетку.

Основная задача адаптивных сеток заключается в увеличении пространственного разрешения в тех местах, где происходят резкие изменения решения (если речь идет о численном решении дифференциальных или интегральных уравнений [64]) или где происходит уплотнение объектов на сцене (если речь идет о компьютерной графике). Существует большое количество вариантов сеток. Интервальные данные представляют собой многомерные прямоугольные параллелепипеды, поэтому здесь рассматриваются ортогональные сетки. Использование деревьев для структурирования ячеек позволяет быстро производить поиск и естественным образом выполнять адаптацию.

Отметим, что аппарат исследования динамических систем достаточно широко развит [65–68] и обобщение его на динамические системы с интервальными параметрами представляет практический интерес.

Первая глава посвящена разработке алгоритма адаптивной интерполяции [55, 56, 61] на основе kd-дерева [69, 70]. В начале главы приводится описание классической интервальной арифметики. Дается постановка интервальной задачи Коши для систем ОДУ. Описывается алгоритм адаптивной интерполяции, формулируются и доказываются утверждения относительно его условий применимости, сходимости и погрешности. Проводится апробация алгоритма на ряде представительных примеров разной размерности, содержащих различное количество интервальных параметров, выполняется сравнение с методом Монте-Карло.

Идея алгоритма заключается в построении динамической структурированной сетки на основе kd-дерева над множеством, образованным интервальными параметрами задачи. Каждая вершина дерева представляет собой интерполяционную сетку, соответствующую заданной степени интерполяционного многочлена. С каждым узлом сетки сопоставляется решение, найденное при параметрах, определяемых положением узла в пространстве. В процессе выполнения алгоритма на каждом шаге интегрирования исходной системы ОДУ строится кусочно-полиномиальная функция, которая интерполирует зависимость решения задачи от точечных значений интервальных параметров.

Каждая итерация алгоритма состоит из двух этапов. На первом этапе все решения, связанные с узлами интерполяционных сеток, переносятся на следующий временной слой с помощью какого-либо численного метода интегрирования. На втором этапе происходит перестроение kd-дерева по принципу минимизации ошибки интерполяции в вершинах.

Во **второй главе** рассматриваются основные аспекты распараллеливания и реализации алгоритма адаптивной интерполяции с применением технологии CUDA [57, 60, 62, 71]. Описываются используемые в алгоритме структуры данных и их особенности с точки зрения параллельной работы. Приводятся результаты вычислительных экспериментов, которые демонстрируют в ряде случаев стократное ускорение по сравнению с вычислениями на центральном процессоре.

В **третьей главе** представлен обзор существующих библиотек и реализованных в них методов моделирования динамических систем с интервальными параметрами. Проведено сравнение полученных результатов с результатами, даваемыми доступными программными библиотеками AWA, VNODE, COSY Infinity, RiOT и FlowStar, которое показывает превосходство разработанного авторами алгоритма и его реализации с точки зрения точности и вычислительных затрат.

В **четвертой главе** приводятся результаты применения программы [62] для решения прикладных и исследовательских задач. Вначале рассматриваются динамические системы, в которых имеют место бифуркации и хаос. По качественному изменению структуры адаптивной сетки определяются режимы, которые возникают в динамической системе.

Далее выполняется расчет трубки траекторий для астероида FX11 [43, 72] в условиях неопределенности положения астероида и его скорости. Производится сравнение полученных результатов с результатами, представленными в других работах.

В конце главы рассматриваются задачи химической кинетики [56, 58] и газовой динамики [59]. Проведено моделирование горения смеси водорода и кислорода при наличии неопределенностей в константах скоростей реакций. Представлена математическая модель, описывающая химические неравновесные течения с неопределенностями в константах скоростей реакций. Приводятся результаты численного

исследования влияния неопределенностей на структуру детонационной волны, а также на параметры установившегося течения, такие, как время задержки воспламенения и концентрация вредных веществ на выходе из сопла.

В **пятой главе** описываются перспективы развития алгоритма адаптивной интерполяции. Рассматриваются различные подходы к уменьшению «проклятия размерности» на основе крестовой аппроксимации, TT-разложения и разреженных сеток. Эти подходы успешно применены как к модельным задачам, так и к прикладным задачам химической кинетики. Выполнено моделирование динамических систем более чем с десятью интервальными параметрами.

ГЛАВА 1. АЛГОРИТМ АДАПТИВНОЙ ИНТЕРПОЛЯЦИИ

1.1. ИНТЕРВАЛЬНАЯ АРИФМЕТИКА

Вначале рассмотрим основу классических интервальных методов — интервальную арифметику. Существует большое количество различных арифметик, различающихся как определением самих интервалов, так и возможными операциями над ними. В данном подразделе приводятся основные определения и свойства классической интервальной арифметики (Б.С. Добронец [20], С.П. Шарый [22]), являющейся базой для построения более сложных арифметик.

При рассмотрении арифметики интервалы выделяются жирным шрифтом, а их множество обозначается через $\mathbb{IR}$:

$$\mathbb{IR} = \left\{ \mathbf{x} = \left[\underline{x}, \overline{x} \right] \mid \underline{x} \le \overline{x}, \underline{x}, \overline{x} \in \mathbb{R} \right\}.$$

Над интервалами определяются арифметические операции — сложение, вычитание, умножение и деление — «по представителям», то есть в соответствии со следующим фундаментальным принципом:

$$\mathbf{a} * \mathbf{b} = \{ a * b \mid a \in \mathbf{a},\ b \in \mathbf{b} \}$$

для интервалов $\mathbf{a}$ и $\mathbf{b}$, таких, что выполнение точечной операции $a * b$, где $* \in \{+, -, \times, /\}$, имеет смысл для любых $a \in \mathbf{a}$ и $b \in \mathbf{b}$. Равносильное определение арифметических операций для интервалов задается следующими формулами:

$$\mathbf{a} + \mathbf{b} = \left[\underline{a} + \underline{b}, \overline{a} + \overline{b} \right],$$

$$\mathbf{a} - \mathbf{b} = \left[\underline{a} - \overline{b}, \overline{a} - \underline{b} \right],$$

$$\mathbf{a} \times \mathbf{b} = \left[\min\left(\underline{a}\underline{b}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b} \right), \max\left(\underline{a}\underline{b}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b} \right) \right],$$

$$\mathbf{a} / \mathbf{b} = \mathbf{a} \times \left[1 / \overline{b}, 1 / \underline{b} \right].$$

Интервальные операции сложения и умножения являются коммутативными и ассоциативными, то есть для $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{IR}$ выполняются соотношения:

$$\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a},$$

$$\mathbf{a} + (\mathbf{b} + \mathbf{c}) = (\mathbf{a} + \mathbf{b}) + \mathbf{c},$$

$$\mathbf{a} \times \mathbf{b} = \mathbf{b} \times \mathbf{a},$$

$$\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = (\mathbf{a} \times \mathbf{b}) \times \mathbf{c}.$$

Если $r(x)$ — непрерывная унарная операция на $\mathbb{R}$, то

$$r(\mathbf{a}) = \left[\min_{x \in \mathbf{a}} (r(x)), \max_{x \in \mathbf{a}} (r(x)) \right].$$

Далее приводятся отличия интервальной арифметики от обычной арифметики. Вместо дистрибутивности умножения относительно сложения выполняется субдистрибутивность

$$\mathbf{a} \times (\mathbf{b} + \mathbf{c}) \subset \mathbf{a} \times \mathbf{b} + \mathbf{a} \times \mathbf{c}.$$

Это свойство означает, что раскрытие скобок в выражениях может приводить к увеличению ширины получающегося результата.

Множество интервалов $\mathbb{IR}$ не является полем, так как элементы $\mathbb{IR}$ не имеют обратных элементов относительно сложения и умножения. В частности, для невырожденного интервала **a** получаем:

$$\mathbf{a} - \mathbf{a} \neq 0, \quad \mathbf{a} / \mathbf{a} \neq 1.$$

Вместо этого действуют два правила сокращения:

$$\mathbf{a} + \mathbf{b} = \mathbf{a} + \mathbf{c} \rightarrow \mathbf{b} = \mathbf{c},$$

$$\mathbf{a} \times \mathbf{b} = \mathbf{a} \times \mathbf{c} \rightarrow \mathbf{b} = \mathbf{c}, \; 0 \notin \mathbf{a}.$$

Приведем несколько характеристик интервалов.

Шириной интервала **a** называется величина

$$\mathrm{wid}(\mathbf{a}) = \overline{a} - \underline{a}.$$

Серединой интервала **a** называется величина

$$\mathrm{mid}(\mathbf{a}) = \frac{1}{2}(\overline{a} + \underline{a}).$$

Радиусом интервала **a** называется величина

$$\mathrm{rad}(\mathbf{a}) = \frac{1}{2}(\overline{a} - \underline{a}).$$

Объединенным расширением вещественной функции $f(x)$ называется интервальная функция $\mathbf{f}_{un}(\mathbf{x})$, такая, что

$$\mathbf{f}_{un}(\mathbf{x}) = \bigcup_{x \in \mathbf{x}} f(x).$$

Интервальным расширением вещественной функции $f(x)$, $x \in D \subset \mathbb{R}^n$ называется интервальная функция $\mathbf{f}$, $x \in \mathbb{R}^n$, такая, что

$$f(x) = \mathbf{f}(x), \forall x \in D.$$

Для любой вещественной рациональной функции $f(x)$, $x \in \mathbb{R}^n$ интервальное расширение можно получить естественным путем, если заменить аргументы x_i и все арифметические операции соответственно интервальными числами и операциями. Построенное таким образом интервальное расширение обозначается $\mathbf{f}_{ne}(\mathbf{x})$. Результат вычисления естественного интервального расширения существенно зависит от способа записи рационального выражения (см. ниже пример 1.1).

В отличие от интервальных расширений, объединенное расширение является оптимальным с точки зрения ширины получаемого результата. Но его построение на практике затруднительно.

Основная теорема интервальной арифметики. Пусть $f(x_1, x_2, ..., x_n)$ — рациональное выражение, в котором каждая переменная встречается не более одного раза и только в первой степени, $\mathbf{f}_{ne}(\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n})$ — его естественное расширение. Тогда

$$\mathbf{f}_{un}(\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}) = \mathbf{f}_{ne}(\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n})$$

для любого набора $(\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n})$, такого, что $\mathbf{f}_{ne}(\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n})$ имеет смысл.

Пример 1.1. Дана вещественная функция:

$$f(x, y) = xy + x + y + 1,\ x \in [0, 1],\ y \in [-1, 0],$$

$$\mathbf{f}(\mathbf{x}, \mathbf{y}) = [0, 1] \times [-1, 0] + [0, 1] + [-1, 0] + 1 = [-1, 2].$$

Преобразуем $f(x, y)$:

$$f(x, y) = xy + x + y + 1 = (x + 1)(y + 1),$$

$$\mathbf{f}(\mathbf{x}, \mathbf{y}) = ([0, 1] + 1) \times ([-1, 0] + 1) = [0, 2].$$

Если удовлетворить условиям теоремы невозможно, нужно хотя бы стремиться к уменьшению числа вхождений каждой переменной. Эффект обертывания, который возникает при использовании интервальных методов, является следствием из основной теоремы. С каждой итерацией алгоритма количество вхождений интервальных переменных увеличивается, что приводит к постоянному увеличению получаемых интервальных оценок решений.

Пример 1.2. Рассмотрим дискретную динамическую систему:

$$x^{k+1} = \frac{1}{2}\begin{pmatrix} 1 & \sqrt{3} \\ -\sqrt{3} & 1 \end{pmatrix} x^k,\ \mathbf{x}^0 = \begin{pmatrix} [-0.1,\ 0.1] \\ 1 + [-0.1,\ 0.1] \end{pmatrix}.$$

Выполним несколько итераций:

$$\mathbf{x}^1 = \frac{1}{2}\begin{pmatrix} \mathbf{x}_1^0 + \sqrt{3}\mathbf{x}_2^0 \\ -\sqrt{3}\mathbf{x}_1^0 + \mathbf{x}_2^0 \end{pmatrix} = \begin{pmatrix} [0.729423,\ 1.00263] \\ [0.363397,\ 0.636603] \end{pmatrix},$$

$$\mathbf{x}^2 = \frac{1}{4}\begin{pmatrix} \mathbf{x}_1^0 + \sqrt{3}\mathbf{x}_2^0 - 3\mathbf{x}_1^0 + \sqrt{3}\mathbf{x}_2^0 \\ -\sqrt{3}\mathbf{x}_1^0 - 3\mathbf{x}_2^0 - \sqrt{3}\mathbf{x}_1^0 + \mathbf{x}_2^0 \end{pmatrix} = \begin{pmatrix} [0.679423,\ 1.05263] \\ [-0.686603,\ -0.313397] \end{pmatrix},$$

$$\mathbf{x}^3 = \frac{1}{8}\begin{pmatrix} \mathbf{x}_1^0 + \sqrt{3}\mathbf{x}_2^0 - 3\mathbf{x}_1^0 + \sqrt{3}\mathbf{x}_2^0 - 3\mathbf{x}_1^0 - 3\sqrt{3}\mathbf{x}_2^0 - 3\mathbf{x}_1^0 + \sqrt{3}\mathbf{x}_2^0 \\ -\sqrt{3}\mathbf{x}_1^0 - 3\mathbf{x}_2^0 + 3\sqrt{3}\mathbf{x}_1^0 - 3\mathbf{x}_2^0 - \sqrt{3}\mathbf{x}_1^0 - 3\mathbf{x}_2^0 - \sqrt{3}\mathbf{x}_1^0 + \mathbf{x}_2^0 \end{pmatrix},$$

$$\mathbf{x}^4 = \ldots,$$

$$\mathbf{x}^k = \begin{pmatrix} \sin\left(k\pi/3\right) \\ \cos\left(k\pi/3\right) \end{pmatrix} + \left(\frac{1+\sqrt{3}}{2}\right)^k \begin{pmatrix} [-0.1,\ 0.1] \\ [-0.1,\ 0.1] \end{pmatrix}. \qquad (1.1)$$

Число вхождений каждой переменной возрастает экспоненциально, что приводит к экспоненциальному росту ширины области. На рис. 1 представлена геометрическая иллюстрация эффекта обертывания, который возникает в данном примере. На каждом шаге полученное множество заменяется большим квадратом со сторонами, параллельными осям координат, в результате чего образуется последовательность вложенных друг в друга областей. Интервальные оценки как бы оборачивают искомое множество, увеличиваясь при этом в размере. Отсюда и название «эффект обертывания».

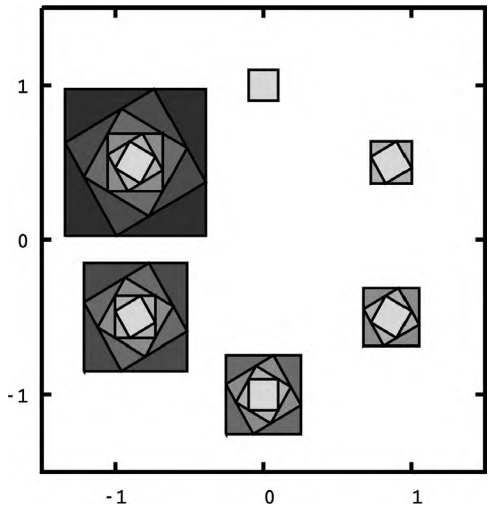

Рис. 1. Иллюстрация эффекта обертывания

Все приведенные выше особенности интервальной арифметики показывают, что ее использование влечет за собой ряд трудностей, зачастую непреодолимых. Поэтому при построении вычислительных схем и алгоритмов желательно как можно меньше использовать в чистом виде интервальные вычисления.

В общем случае для подавления эффекта обертывания необходимо устанавливать функциональные зависимости между интервальными параметрами и решением в каждый момент времени. Именно эта идея лежит в основе работ М. Берза [33–38] и А.Н. Рогалева [23–25]. Разработанные ими методы направлены в первую очередь на получение гарантированных оценок решений, в то время как рассматриваемый далее алгоритм направлен на высокую робастность, универсальность и быстроту.

Для последней задачи (пример 1.2) функциональная зависимость имеет следующий вид:

$$x^k = \begin{pmatrix} \cos\left(\frac{k\pi}{3}\right) & \sin\left(\frac{k\pi}{3}\right) \\ -\sin\left(\frac{k\pi}{3}\right) & \cos\left(\frac{k\pi}{3}\right) \end{pmatrix} x^0 .$$

Отметим, что она существенно отличается от полученной ранее интервальной оценки $\mathbf{x}^k$ (1.1).

1.2. ПОСТАНОВКА ЗАДАЧИ

Интервальная постановка задачи Коши:

$$\begin{cases} u' = f(t, u, \eta), \\ u(0) = u_0 \in \mathbf{u}_0, \eta \in \boldsymbol{\eta}, \\ t \in [0, t_N], \end{cases} \tag{1.2}$$

где $f : \mathbb{R} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_\eta} \to \mathbb{R}^{n_u}$ — правая часть системы ОДУ, $\mathbf{u}_0 \in \mathbb{IR}^{n_u}$ — интервальные начальные условия; $\boldsymbol{\eta} \in \mathbb{IR}^{n_\eta}$ — интервальные параметры. Вектор-функция f удовлетворяет условиям, обеспечивающим существование и единственность решения при всех $u_0 \in \mathbf{u}_0$ и $\eta \in \boldsymbol{\eta}$.

Для дальнейшего удобства описания алгоритма преобразуем (1.2) к автономной системе ОДУ без параметров путем добавления в систему фиктивных уравнений, а также сгруппируем интервальные начальные условия:

$$\begin{cases} y' = f(y), \\ y(0) = y_0 \in \mathbf{y}_0, \\ t \in [0, t_N], \end{cases} \tag{1.3}$$

где $f : \mathbb{R}^n \to \mathbb{R}^n$, $\mathbf{y}_0 = \left(\mathbf{y}_0^1, \mathbf{y}_0^2, ..., \mathbf{y}_0^m, y_0^{m+1}, ..., y_0^n\right)^T$, $n = n_u + n_\eta + 1$.
Цель алгоритма — для каждого момента времени t_k построить кусочно-полиномиальную функцию $P_k(y_0)$, которая интерполирует решение системы (1.3) с контролируемой точностью, и определить ее интервальную оценку.

1.3. ОПИСАНИЕ АЛГОРИТМА

Обозначим через $y_k = y(y_0, t_k)$ решение системы (1.3) в момент t_k. Над множеством, образованным интервальными начальными условиями, построим равномерную регулярную интерполяционную сетку G_k^0, соответствующую корневой вершине дерева и представляющую собой табличную функцию:

$$G_k^0 = \left\{\left(y_0^{i_1 i_2 \ldots i_m}, y_k^{i_1 i_2 \ldots i_m}\right) \mid 0 \le i_l \le p, 1 \le l \le m\right\},$$

$$y_0^{i_1 i_2 \ldots i_m} = \left(\underline{y}_0^1 + \frac{\mathrm{wid}\left(\mathbf{y}_0^1\right)}{p} i_1, \underline{y}_0^2 + \frac{\mathrm{wid}\left(\mathbf{y}_0^2\right)}{p} i_2, \ldots, \underline{y}_0^m + \frac{\mathrm{wid}\left(\mathbf{y}_0^m\right)}{p} i_m, y_0^{m+1}, \ldots, y_0^n\right)^T,$$

$$y_k^{i_1 i_2 \ldots i_m} = y\left(y_0^{i_1 i_2 \ldots i_m}, t_k\right),$$

где p — степень интерполяционного многочлена по каждой переменной. Отметим, что требование равномерности разбиения необязательно и используется здесь для сокращения записи. В каждом конкретном случае параметр p подбирается из соображений устойчивости и минимизации вычислительных затрат: увеличение p приводит к улучшению точности, ухудшению устойчивости и увеличению количества вычислений в рамках одной вершины.

Для каждого $y_k^{i_1 i_2 \ldots i_m}$ вычислим $y_{k+1}^{i_1 i_2 \ldots i_m} = y\left(y_0^{i_1 i_2 \ldots i_m}, t_{k+1}\right) = y\left(y_k^{i_1 i_2 \ldots i_m}, t_{k+1} - t_k\right)$, решив соответствующую неинтервальную систему ОДУ:

$$\begin{cases} y' = f(y), \\ y(0) = y_k^{i_1 i_2 \ldots i_m}, \\ t \in [0, t_{k+1} - t_k]. \end{cases}$$

По полученной табличной функции

$$G_{k+1}^0 = \left\{\left(y_0^{i_1 i_2 \ldots i_m}, y_{k+1}^{i_1 i_2 \ldots i_m}\right) \mid 0 \le i_l \le p, 1 \le l \le m\right\}$$

построим интерполяционный многочлен $P_{k+1}^0(y)$, например в форме Лагранжа:

$$P_{k+1}^0(y) = \sum_{i_1=0}^{p} \ldots \sum_{i_m=0}^{p} y_{k+1}^{i_1 i_2 \ldots i_m} l^{i_1 i_2 \ldots i_m}(y),$$

где $l^{i_1 i_2 \ldots i_m}(y)$ — базисные полиномы Лагранжа [73]:

$$l^{i_1 i_2 \ldots i_m}(y) = \prod_{j=0}^{p} \prod_{\substack{l=0 \\ i_l \ne j}}^{m} \frac{p \dfrac{y^l - \underline{y}_0^l}{\mathrm{wid}\left(\mathbf{y}_0^l\right)} - j}{i_l - j}.$$

Если апостериорная погрешность интерполяции

$$error = \max_{y_0 \in \mathbf{y}_0} \left\| y(y_0, t_{k+1}) - P_{k+1}^0(y_0) \right\| \tag{1.4}$$

больше некоторого заданного числа ε, то G_k^0 разбивается на две сетки — G_k^1 и G_k^2 — таким образом, чтобы их оценка погрешности интерполяции была меньше *error*. Для них выполняются все те же самые действия, что и для сетки G_k^0, и при необходимости они тоже разбиваются. Если $y(y_0, t_{k+1})$ непрерывно дифференцируема $p + 1$ раз по y_0, то можно показать, что процесс дробления конечен. В общем случае с теоретической точки зрения погрешность интерполяции оценивается сверху через старшие производные с коэффициентами пропорциональности, которые характеризуют интерполяционную сетку и ее размер.

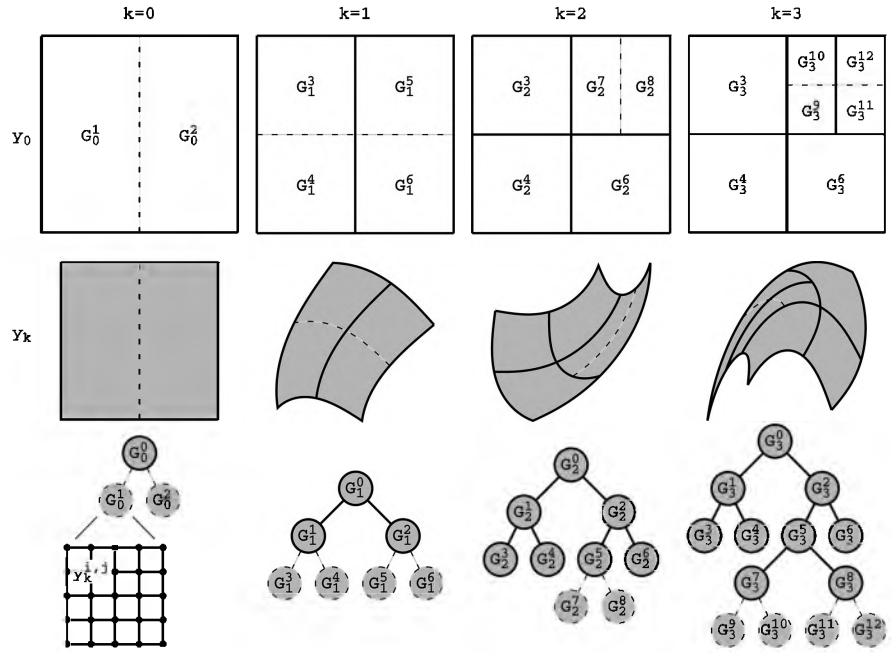

Рис. 2. Иллюстрация работы алгоритма

При каждом дроблении эти коэффициенты уменьшаются, и, как следствие, уменьшается и погрешность интерполяции. На практике определяется минимальный размер ячейки, обычно сопоставимый с

машинным эпсилоном, по достижении которого дробление дальше не происходит, а соответствующая область пространства помечается как «область разрыва».

В результате на момент t_{k+1} будет получено kd-дерево и соответствующая ему кусочно-полиномиальная функция, интерполирующая решение с заданной точностью. Процесс построения kd-дерева проиллюстрирован на рис. 2. Строить с нуля на каждом шаге kd-дерево нет необходимости, вместо этого используется дерево, полученное на предыдущем шаге, и в зависимости от оценки погрешности интерполяции оно перестраивается. Процесс дробления вершин всегда происходит на предыдущем шаге (пунктирные линии), потому что при создании новых вершин выполняется интерполяция связанных с их узлами значений, которую необходимо выполнять в момент, когда погрешность еще допустима. Если для вершины и всех ее потомков ошибка интерполяции становится приемлемой, то потомки удаляются и сама вершина становится листом.

1.4. АПОСТЕРИОРНАЯ ОЦЕНКА ПОГРЕШНОСТИ ИНТЕРПОЛЯЦИИ

Рассмотрим более подробно апостериорную оценку погрешности интерполяции для одной вершины. Обозначим через $\mathbf{d}^i \subset \mathbf{y}_0$ интервальный вектор, который соответствует i-й вершине дерева. На практике оценка погрешности интерполяции (1.4) выполняется не для всей области $\mathbf{d}^i$, а только для некоторых точек из нее. Выделим два подхода к их выбору. Первый заключается в добавлении случайным образом тестового множества точек $V_k^{t,i}$ в каждую вершину при ее создании (рис. 3,*а*):

$$V_k^{t,i} = \left\{ \left(y_0^{t,i}, y_k^{t,i} \right) \mid y_k^{t,i} = y\left(y_0^{t,i}, t_k \right) \right\},$$

$$y_0^{t,i} = \left(\text{rand}\left[\mathbf{d}^{i,1} \right], \text{rand}\left[\mathbf{d}^{i,2} \right], ..., \text{rand}\left[\mathbf{d}^{i,m} \right], y_0^{m+1}, ..., y_0^n \right)^T,$$

$$\text{rand}\left[\mathbf{d}^{i,j} \right] = \text{rand}\left[\underline{d}^{i,j}, \overline{d}^{i,j} \right], \; j = \overline{1, m}.$$

Значение $y_k^{t,i}$ в момент создания вершины вычисляется путем интерполяции. При перевычислении решения на следующий вре-

менной слой все элементы, содержащиеся в $V_k^{i,t}$, также переносятся на $k + 1$ слой:

$$V_k^{t,i} \to V_{k+1}^{t,i} = \left\{ \left(y_0^{t,i}, y_{k+1}^{t,i} \right) \mid y_{k+1}^{t,i} = y\left(y_0^{t,i}, t_{k+1} \right) = y\left(y_k^{t,i}, t_{k+1} - t_k \right), \left(y_0^{t,i}, y_k^{t,i} \right) \in V_k^{t,i} \right\}.$$

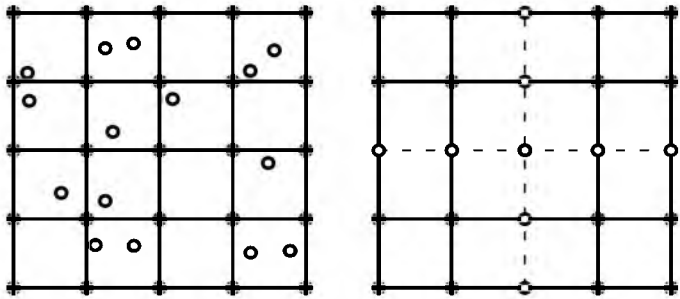

Рис. 3. Выбор точек, в которых выполняется оценка погрешности интерполяции

С учетом введенных обозначений оценка погрешности интерполяции (1.4) записывается следующим образом:

$$error = \max_{\left(y_0^{t,i}, y_{k+1}^{t,i} \right) \in V_{k+1}^{t,i}} \left\| y_{k+1}^{t,i} - P_{k+1}^i \left(y_0^{t,i} \right) \right\|.$$

Определим относительную погрешность с помощью нормировки по «точному» значению:

$$error = \max_{\left(y_0^{t,i}, y_{k+1}^{t,i} \right) \in V_{k+1}^{t,i}} \frac{\left\| y_{k+1}^{t,i} - P_{k+1}^i \left(y_0^{t,i} \right) \right\|}{\left\| y_{k+1}^{t,i} \right\|}, \left\| y_{k+1}^{t,i} \right\| \neq 0.$$

В задачах, где решения в зависимости от интервальных параметров различаются на порядки, имеет смысл выполнять нормировку не по «точному» значению, а по максимальному:

$$error = \max_{\left(y_0^{t,i}, y_{k+1}^{t,i} \right) \in V_{k+1}^{t,i}} \frac{\left\| y_{k+1}^{t,i} - P_{k+1}^i \left(y_0^{t,i} \right) \right\|}{\max\limits_j \max\limits_{y_{k+1}^{t,j} \in V_{k+1}^{t,j}} \left\| y_{k+1}^{t,j} \right\|}, \max_j \max_{y_{k+1}^{t,j} \in V_{k+1}^{t,j}} \left\| y_{k+1}^{t,j} \right\| \neq 0. \quad (1.5)$$

Так как значения $y_0^{t,i}$, входящие в тестовое множество, не изменяются со временем, то возможно заранее вычислить значения базисных полиномов Лагранжа

$$l^{i_1 i_2 \ldots i_m}\left(y_0^{t,i}\right) = \prod_{j=0}^{p} \prod_{\substack{l=0 \\ i_l \neq j}}^{m} \frac{p\dfrac{y_0^{t,i,l} - \underline{d}^{i,l}}{\mathrm{wid}\left(\mathbf{d}^{i,l}\right)} - j}{i_l - j}$$

и использовать их на каждом шаге.

Второй подход основан на понижении порядка интерполяции: из сетки убираются некоторые узлы, значения в которых интерполируются по оставшимся узлам с пониженным порядком (рис. 3,*б*). Как и в первом подходе, здесь предвычисляются значения базисных полиномов Лагранжа. Если в каждой вершине относительное расположение узлов интерполяции одинаково, то они будут одинаковыми для всех вершин.

С учетом того что оценка погрешности во втором случае является завышенной, она требует меньшего количества вычислительных ресурсов, поэтому предпочтение отдается ей. На рис. 4 показана геометрическая интерпретация погрешности интерполяции в евклидовой норме для одной вершины. Все линии получены путем интерполяции по закрашенным точкам и соответствуют линиям на рис. 3,*б*.

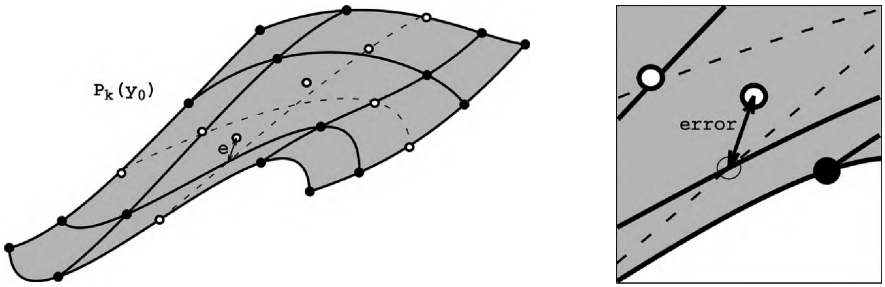

Рис. 4. Геометрическая интерпретация погрешности интерполяции для одной вершины

Для оценки глобальной погрешности используется метод Монте-Карло. В начале работы алгоритма создается тестовое множество

$$V_0^t = \left\{\left(y_0^t,\, y_0^t\right)\right\},\; y_0^t = \left(\text{rand}\left[\mathbf{y}_0^1\right],\, \text{rand}\left[\mathbf{y}_0^2\right],\, ...,\, \text{rand}\left[\mathbf{y}_0^m\right],\, y_0^{m+1},\, ...,\, y_0^n\right)^T,$$

содержащее достаточно много случайных значений интервальных параметров. Все действия выполняются точно так же, как в первом подходе, только не на уровне одной вершины, а на уровне всего kd-дерева. В момент, когда происходит перемещение дерева на следующий временной слой, все значения, содержащиеся в тестовом множестве, обновляются. Далее для каждой точки выполняются поиск по дереву той интерполяционной сетки, в которую она попадает, и вычисление погрешности. Стоит отметить, что поиск не является трудозатратной операцией, так как число переходов по дереву, которые необходимо совершить, ограничено сверху высотой дерева.

1.5. РАЗБИЕНИЕ ВЕРШИН KD-ДЕРЕВА

При расщеплении вершины на две на предыдущем шаге выполняется построение различных вариантов ее разбиения (рис. 5), из которых и выбирается оптимальный вариант с точки зрения погрешности интерполяции на текущем шаге.

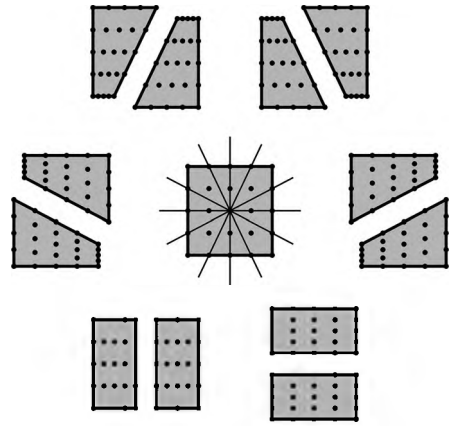

Рис. 5. Различные варианты разбиения вершины

Формально на способы разбиения вершины не накладываются никакие ограничения, кроме того, что всегда нужно оставаться в

рамках интерполяции. С практической точки зрения разбиение целесообразно производить с помощью гиперплоскости, перпендикулярной одной из координатных осей. Это позволит работать только с прямоугольными многомерными параллелепипедами и избежать рассмотрения множества частных случаев.

При дроблении вершин всегда на равные части и использовании равномерных интерполяционных сеток узлы в дочерних вершинах полностью дублируют узлы в родительских вершинах, что, во-первых, приводит к уменьшению вычислительных затрат вдвое, а во-вторых, вдвое уменьшает количество интерполируемых значений в процессе работы алгоритма и, как следствие, улучшает точность. Поэтому остановимся на таком способе дробления и использовании равномерных сеток. В этом случае для выполнения одного разбиения потребуется перебор m различных вариантов.

Предположим, что s-ю вершину

$$G_k^s = \left\{ \left(y_0^{s,\, i_1 i_2 \ldots i_m},\, y_k^{s,\, i_1 i_2 \ldots i_m} \right) \mid 0 \le i_l \le p, 1 \le l \le m \right\},$$

$$y_0^{s,\, i_1 i_2 \ldots i_m} = \left(\underline{d}^{s,\,1} + \frac{wid\left(\mathbf{d}^{s,\,1}\right)}{p} i_1, \ldots, \underline{d}^{s,\,m} + \frac{wid\left(\mathbf{d}^{s,\,m}\right)}{p} i_m, y_0^{m+1}, \ldots, y_0^n \right)^T$$

требуется разбить на две: a и b. Рассмотрим j-й вариант разбиения:

$$j: \begin{cases} G_k^a = \left\{ \left(y_0^{a,\, i_1 i_2 \ldots i_m},\, y_k^{a,\, i_1 i_2 \ldots i_m} \right) \mid 0 \le i_l \le p, 1 \le l \le m \right\}, \\ G_k^b = \left\{ \left(y_0^{b,\, i_1 i_2 \ldots i_m},\, y_k^{b,\, i_1 i_2 \ldots i_m} \right) \mid 0 \le i_l \le p, 1 \le l \le m \right\}, \end{cases}$$

$$y_0^{a,\, i_1 i_2 \ldots i_m} = \left(\ldots, \underline{d}^{s,\, j} + \frac{wid\left(\mathbf{d}^{s,\, j}\right)}{2p} i_j, \ldots \right)^T,$$

$$y_0^{b,\, i_1 i_2 \ldots i_m} = \left(\ldots, \underline{d}^{s,\, j} + \frac{wid\left(\mathbf{d}^{s,\, j}\right)}{2} + \frac{wid\left(\mathbf{d}^{s,\, j}\right)}{2p} i_j, \ldots \right)^T.$$

Значения $y_k^{a,\, i_1 i_2 \ldots i_m}$ и $y_k^{b,\, i_1 i_2 \ldots i_m}$ вычисляются путем интерполяции по сетке G_k^s. При этом точно так же, как было при вычислении погрешности интерполяции, здесь заранее вычисляются все базисные полиномы Лагранжа, необходимые при создании новых вершин.

Далее полученные сетки «переносятся» на $k + 1$ временной слой. Для каждого $y_k^{a,\, i_1 i_2 \ldots i_m}$ и $y_k^{b,\, i_1 i_2 \ldots i_m}$ вычисляется $y_{k+1}^{a,\, i_1 i_2 \ldots i_m} = y\left(y_k^{a,\, i_1 i_2 \ldots i_m}, t_{k+1} - t_k\right)$ и $y_{k+1}^{b,\, i_1 i_2 \ldots i_m} = y\left(y_k^{b,\, i_1 i_2 \ldots i_m}, t_{k+1} - t_k\right)$ путем интегрирования соответствующих неинтервальных систем ОДУ. Для сеток вычисляется погрешность интерполяции $error_j^a$ и $error_j^b$. Варианту ставится в соответствие число z_j, равное количеству сеток, для которых погрешность больше заданного числа ε.

Выбор между вариантами j_1 и j_2 осуществляется следующим образом. Если $z_{j_1} < z_{j_2}$, то оставляется вариант j_1. Если $z_{j_1} > z_{j_2}$, то оставляется вариант j_2. Если $z_{j_1} = z_{j_2}$ и $\max\left(error_{j_1}^a, error_{j_1}^b\right) < \max\left(error_{j_2}^a, error_{j_2}^b\right)$, то оставляется вариант j_1, иначе — j_2.

Так как оценка погрешности выполняется только для определенных точек из соответствующей области, то может возникнуть ситуация, когда разбиения будут происходить некорректно. В связи с этим при выборе варианта разбиения дополнительно используются весовые коэффициенты, которые характеризуют вытянутость области, соответствующей данной вершине. Эти коэффициенты вычисляются из принципа, что дробить параллелепипед нужно гиперплоскостью, перпендикулярной самому длинному его измерению. Полученные погрешности $error_j^a$ и $error_j^b$ умножаются на отношение изначальной длины параллелепипеда по j-му измерению к текущей длине $\dfrac{\mathrm{wid}\left(\mathbf{y}_0^j\right)}{\mathrm{wid}\left(\mathbf{d}^{s,j}\right)}$.

1.6. ПОСТРОЕНИЕ ИНТЕРВАЛЬНОГО РЕШЕНИЯ

После того как в конкретный момент времени t_k получено kd-дерево и, следовательно, получена кусочно-полиномиальная функция $\tilde{y}(y_0): \mathbb{R}^m \to \mathbb{R}^n$, необходимо определить ее область значений, это и будет интервальная оценка решения. Для этого надо решить $2n$ задач на поиск экстремума:

$$\mathbf{y} = \left(\left[\min_{y_0 \in \mathbf{y}_0} \tilde{y}^1(y_0),\ \max_{y_0 \in \mathbf{y}_0} \tilde{y}^1(y_0)\right], \ldots, \left[\min_{y_0 \in \mathbf{y}_0} \tilde{y}^n(y_0),\ \max_{y_0 \in \mathbf{y}_0} \tilde{y}^n(y_0)\right]\right).$$

Для решения данных задач существуют как классические методы [74], так и интервальные [75–77]. Классические методы в общем слу-

чае не могут гарантировать, что найденное решение является лучшим. Большую роль в них играет стартовое значение, с которого начинается поиск. В данной задаче в качестве стартового значения берется лучшая найденная точка среди всех точек, соответствующих узлам интерполяционных сеток, что является хорошим приближением. Также здесь применимы методы первого и второго порядка, так как функция дифференцируема (в рамках одной вершины kd-дерева).

Основное преимущество интервальных методов для решения задач на поиск экстремума заключается в гарантии того, что в найденной интервальной оценке точно содержится оптимальное решение. Но для этих методов характерна более высокая вычислительная сложность и в некоторых случаях требуется дополнительная память.

В целом для вычисления **y** можно использовать практически любые существующие методы, так как процесс вычисления функции $\tilde{y}(y_0)$ не ресурсоемкий, и поэтому этот этап алгоритма по вычислительным затратам несущественен по сравнению с остальными этапами.

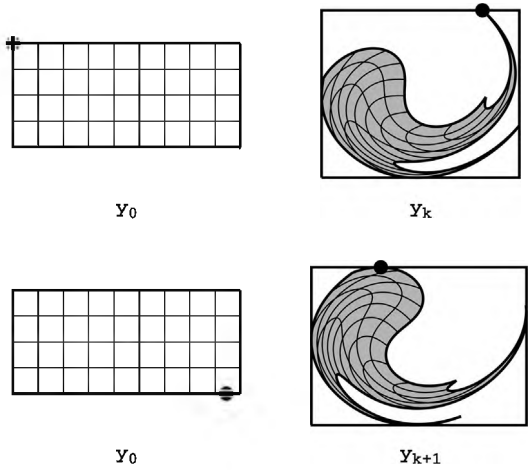

Рис. 6. Перескакивание точки экстремума

Вообще говоря, можно было бы использовать методы оптимизации для нахождения интервальных оценок решений систем ОДУ, при этом понимая под оптимизируемой функцией решатель ОДУ. Однако такой подход малоэффективен. Во-первых, в каждый момент времени

в процессе поиска минимума или максимума придется многократно решать систему ОДУ, что по вычислительным затратам дорого. Во-вторых, классические методы не гарантируют, что найден глобальный на интервале минимум или максимум (так как функция является «черным ящиком»). Зачастую траектория точки экстремума на множестве $\mathbf{y}_0$ является разрывной (рис. 6), поэтому нельзя использовать в качестве стартового значения предыдущее найденное значение. А интервальные методы потребуют интегрирования интервальных систем ОДУ, что приведет к этой же проблеме. В-третьих, полученная интервальная оценка не будет нести информации о структуре множества. Однако для определения «точной» интервальной оценки в конкретный момент времени для последующего сравнения интервальных методов такой подход допустим.

1.7. ОБОСНОВАНИЕ АЛГОРИТМА

Рассмотрим интервальную задачу Коши для одного дифференциального уравнения:

$$\begin{cases} y' = f(y), \\ y(0) = y_0 \in \mathbf{y_0}, \\ t \in [0, t_N]. \end{cases} \tag{1.6}$$

Утверждение 1. *Если правая часть системы (1.6) непрерывно дифференцируема $p + 1$ раз, то для каждого фиксированного t_k можно построить кусочно-полиномиальную функцию степени p, которая интерполирует решение $y(y_0, t_k)$ с заданной точностью.*

Утверждение 2. *Оценка глобальной погрешности алгоритма прямо пропорциональна высоте kd-дерева.*

Доказательство.

Из теоремы о дифференцируемости решения системы ОДУ по начальному условию следует, что решение системы (1.6) $y(y_0, t_k)$ непрерывно дифференцируемо $p + 1$ раз по y_0 [78].

Введем обозначение:

$$M_k^p = \frac{1}{4(p+1)} \max_{\theta \in \mathbf{y}_0} \left| \frac{\partial^{p+1} y(y_0, t_k)}{\partial y_0^{p+1}} \bigg|_{y_0 = \theta} \right|.$$

Построим интерполяционную сетку G_k^0 на отрезке $\left[\underline{y}_0, \overline{y}_0\right]$, представляющую собой табличную функцию:

$$G_k^0 = \left\{\left(y_0^{0,0}, y_k^{0,0}\right), \left(y_0^{0,1}, y_k^{0,1}\right), ..., \left(y_0^{0,p}, y_k^{0,p}\right)\right\},$$

$$\underline{y}_0 = y_0^{0,0} < y_0^{0,1} < ... < y_0^{0,p} = \overline{y}_0,$$

$$y_k^{0,j} = y\left(y_0^{0,j}, t_k\right),\ j = \overline{0,\ p}.$$

Для каждого $y_k^{0,j}$ вычислим значение $y_{k+1}^{0,j} = y\left(y_0^{0,j}, t_{k+1}\right) = y\left(y_k^{0,j}, t_{k+1} - t_k\right)$, решив соответствующую неинтервальную систему ОДУ. По полученной табличной функции

$$G_{k+1}^0 = \left\{\left(y_0^{0,0}, y_{k+1}^{0,0}\right), \left(y_0^{0,1}, y_{k+1}^{0,1}\right), ..., \left(y_0^{0,p}, y_{k+1}^{0,p}\right)\right\}$$

построим интерполяционный многочлен $P_{k+1}^0(y_0)$ степени p. Для приближенного решения

$$\tilde{y}(y_0, t_{k+1}) = P_{k+1}^0(y_0),\ y_0 \in \left[y_0^{0,0}, y_0^{0,p}\right]$$

выполняется оценка [79]:

$$\left|y(y_0, t_{k+1}) - \tilde{y}(y_0, t_{k+1})\right| = \left|\frac{1}{(p+1)!}\frac{\partial^{p+1} y(y_0, t_{k+1})}{\partial y_0^{p+1}}\right|_{y_0=\theta}\prod_{j=1}^{p}\left(y_0 - y_0^{0,j}\right)\le M_{k+1}^p h_0^{p+1},$$

где $\theta \in \left[y_0^{0,0}, y_0^{0,p}\right]$, $h_0 = \max\limits_{j=\overline{1,\ p}}\left[y_0^{0,j} - y_0^{0,j-1}\right]$.

Если величина $M_{k+1}^p h_0^{p+1}$ больше некоторого заданного ε, то сетка G_k^0 разбивается на две сетки:

$$G_k^1 = \left\{\left(y_0^{1,0}, y_k^{1,0}\right), \left(y_0^{1,1}, y_k^{1,1}\right), ..., \left(y_0^{1,p}, y_k^{1,p}\right)\right\},$$

$$G_k^2 = \left\{\left(y_0^{2,0}, y_k^{2,0}\right), \left(y_0^{2,1}, y_k^{2,1}\right), ..., \left(y_0^{2,p}, y_k^{2,p}\right)\right\},$$

$$\underline{y}_0 = y_0^{1,0} < y_0^{1,1} < ... < y_0^{1,p} = y_0^{2,0} < y_0^{2,1} < ... < y_0^{2,p} = \overline{y}_0,$$

$$y_k^{i,j} = y\left(y_0^{i,j}, t_k\right),\ i = 1, 2,\ j = \overline{0,\ p}.$$

Интерполяционные узлы $y_0^{i,j}$ выбираются так, чтобы выполнялось условие

$$h_i = \max_{j=\overline{1,\ p}}\left[y_0^{i,j} - y_0^{i,j-1}\right] < h_0,\ i = 1, 2.$$

Вычислим сетки

$$G_{k+1}^1 = \left\{\left(y_0^{1,0}, y_{k+1}^{1,0}\right), \left(y_0^{1,1}, y_{k+1}^{1,1}\right), ..., \left(y_0^{1,p}, y_{k+1}^{1,p}\right)\right\}$$

и

$$G_{k+1}^2 = \left\{\left(y_0^{2,0}, y_{k+1}^{2,0}\right), \left(y_0^{2,1}, y_{k+1}^{2,1}\right), ..., \left(y_0^{2,p}, y_{k+1}^{2,p}\right)\right\}$$

и построим соответствующие интерполяционные полиномы $P_{k+1}^1(y_0)$ и $P_{k+1}^2(y_0)$. Новое приближенное решение $\tilde{y}(y_0, t_{k+1})$ задается равенством

$$\tilde{y}(y_0, t_{k+1}) = \begin{cases} P_{k+1}^1(y_0), y_0 \in \left[y_0^{1,0}, y_0^{1,p}\right] \\ P_{k+1}^2(y_0), y_0 \in \left[y_0^{2,0}, y_0^{2,p}\right] \end{cases},$$

и оценка погрешности имеет вид:

$$\left|y(y_0, t_{k+1}) - \tilde{y}(y_0, t_{k+1})\right| \le M_{k+1}^{p,i} h_i^{p+1}, \; y_0 \in \left[y_0^{i,0}, y_0^{i,p}\right], \; i = 1, 2,$$

где

$$M_{k+1}^{p,i} = \frac{1}{4(p+1)} \max_{\theta \in \left[y_0^{i,0}, y_0^{i,p}\right]} \left| \frac{\partial^{p+1} y(y_0, t_{k+1})}{\partial y_0^{p+1}} \bigg|_{y_0 = \theta} \right| \le M_{k+1}^p .$$

Если для некоторого i оценка погрешности $M_{k+1}^{p,i} h_i^{p+1} > \varepsilon$, то сетка G_k^i разбивается на две части, и весь процесс начинается сначала, так же, как было при дроблении G_k^0.

Процесс дробления конечен вследствие неравенства $h_i < h_0$, из которого следует уменьшение величины оценки погрешности интерполяции при каждом разбиении. Отметим, что если дробить сетки ровно пополам, сохраняя при этом относительное расположение интерполяционных узлов, то погрешность интерполяции с каждым разбиением уменьшается как минимум в 2^{p+1} раз.

Обозначим через i_L сетки, находящиеся в вершинах-листьях дерева. Тогда решение запишется следующим образом:

$$y(y_0, t_{k+1}) = P_{k+1}^{i_L}(y_0) + \delta_{i_L}, \; y_0 \in \left[y_0^{i_L,0}, y_0^{i_L,p}\right],$$

$$\left|\delta_{i_L}\right| \le \varepsilon, \; \delta_{i_L} = O\left(h_{i_L}^{p+1}\right), \; h_{i_L} = \max_{j=1,p}\left[y_0^{i_L,j} - y_0^{i_L,j-1}\right].$$

Для оценки глобальной погрешности проследим путь получения решения для некоторой точки y_0 до конечного момента интегрирования. Пусть решение в точке y_0 в момент t_N интерполируется по сетке $G_N^{i_L}$, находящейся в листовой вершине kd-дерева. Очевидно, данная сетка была получена путем дробления сетки $G_{k_1}^{i_1}$, находящейся в родительской вершине в некоторый момент t_{k_1}. Она, в свою очередь, получена путем дробления $G_{k_2}^{i_2}$, и так далее, вплоть до сетки $G_{k_d}^0$, где d — глубина вершины, содержащей сетку $G_N^{i_L}$ (рис. 7). То есть для каждого конкретного значения y_0 количество шагов, в которых возникала локальная погрешность от интерполяции, равняется глубине соответствующей вершины и ограничено сверху высотой kd-дерева.

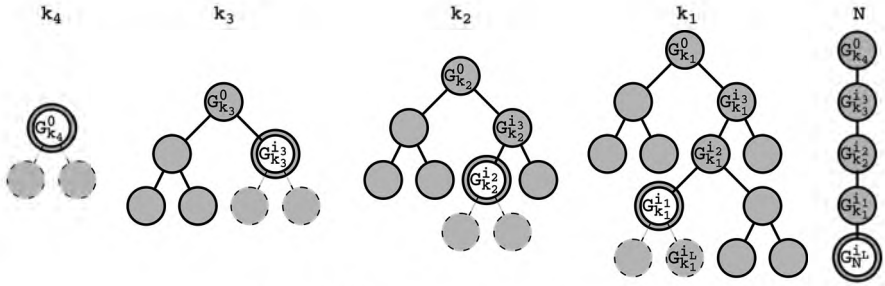

Рис. 7. Путь получения решения

Одним из способов оценки глобальной погрешности является суммирование всех оценок для локальных погрешностей, перенесенных на конец интервала интегрирования [80]. Так как $y(y_0, t)$ удовлетворяет условию Липшица, то справедлива следующая оценка для перенесенной локальной погрешности с k_l-го шага:

$$\left| y\left(\tilde{y}\left(y_0, t_{k_l}\right) + \delta, t_N - t_{k_l}\right) - y\left(\tilde{y}\left(y_0, t_{k_l}\right), t_N - t_{k_l}\right)\right| \le L|\delta|,$$

$$y_0 \in \left[\underline{y}_0, \overline{y}_0\right],$$

где $\tilde{y}\left(y_0, t_{k_l}\right)$ — найденное решение; δ — локальная погрешность; L — константа Липшица.

Рассмотрим две интерполяционные сетки — $\tilde{G}_N^{i_L}$ и $G_N^{i_L}$:

$$\tilde{G}_N^{i_L} = \left\{\left(y_0^{j_L, 0}, y_N^{j_L, 0}\right), \left(y_0^{j_L, 1}, y_N^{j_L, 1}\right), ..., \left(y_0^{j_L, p}, y_N^{j_L, p}\right)\right\},$$

$$G_N^{i_L} = \left\{\left(y_0^{i_L,0}, y_N^{i_L,0} + L\delta_0\right), \left(y_0^{i_L,1}, y_N^{i_L,1} + L\delta_1\right), ..., \left(y_0^{i_L,p}, y_N^{i_L,p} + L\delta_p\right)\right\},$$

$$\left|\delta_j\right| \le |\delta| \le \varepsilon,\ j = \overline{0,\ p}$$

и построим интерполяционные полиномы $\tilde{P}_N^{i_L}(y_0)$ и $P_N^{i_L}(y_0)$. Перенесенную локальную погрешность k_l-го шага можно оценить следующим образом:

$$\left|\tilde{P}_N^{i_L}(y_0) - P_N^{i_L}(y_0)\right| \le KL(p+1)\varepsilon,$$

$$K = \max_{\substack{y_0 \in \left[y_0^{i_L,0}, y_0^{i_L,p}\right] \\ j=\overline{0,\ p}}} \prod_{\substack{l=0 \\ l \ne j}}^{p} \frac{\left|y_0 - y_0^{i_L,l}\right|}{\left|y_0^{i_L,j} - y_0^{i_L,l}\right|},$$

где величина K характеризует сетку и зависит только от относительного расположения узлов интерполяции.

Оценим максимальную высоту дерева. Исходя из оценки локальной погрешности интерполяции, имеем неравенство:

$$M_N^p \left(\frac{wid(\mathbf{y}_0)}{\alpha^d}\right)^{p+1} \le \varepsilon,$$

где d — высота дерева, а число $\alpha > 0$ характеризует пропорцию, в которой дробятся сетки (в случае дробления сеток всякий раз пополам $\alpha = 2$). При достижении требуемой точности процесс дробления заканчивается, и имеем равенство:

$$d = \left\lceil \log_\alpha\left(\mathrm{wid}\left(\mathbf{y}_0\right)\right) + \frac{1}{p+1}\log_\alpha \frac{M_N^p}{\varepsilon} \right\rceil.$$

Просуммировав все перенесенные локальные погрешности, получим оценку глобальной погрешности:

$$\left\lceil \log_\alpha\left(\mathrm{wid}\left(\mathbf{y}_0\right)\right) + \frac{1}{p+1}\log_\alpha \frac{M_N^p}{\varepsilon} \right\rceil KL(p+1)\varepsilon,$$

где числа K и p характеризуют интерполяционные сетки, число α — способ дробления сеток, числа $M_N^p, L, \mathbf{y}_0$ — систему ОДУ, а ε — заранее задаваемая локальная погрешность. При фиксированных параметрах задачи и сеток имеем оценку $O(\varepsilon \ln \varepsilon)$.

Оба утверждения доказаны.

Утверждение 1 возможно усилить с помощью теоремы Вейерштрасса о равномерном приближении непрерывной функции многочленами. Действительно, если потребовать от решения только непрерывность, то его можно приблизить многочленами Бернштейна. Но на практике: если искомая функция хотя бы частично непрерывно-дифференцируема некоторое количество раз, то скорость сходимости здесь будет намного хуже, чем при использовании классической интерполяции, и для достижения такой же точности потребуется на порядки больше узлов.

Для практической оценки трудозатрат определим критерий, который характеризует количество решенных неинтервальных ОДУ (1.3) в процессе работы алгоритма:

$$I = \frac{1}{t_N} \sum_{k=1}^{N} \left(N_i(t_k) + (m-1) N_{new}(t_k) \right) (t_k - t_{k-1}), \qquad (1.7)$$

где $N_i(t_k)$ — количество узлов в сетках без повторений в момент t_k, $N_{new}(t_k)$ — количество новых узлов, созданных в процессе перестроения kd-дерева, множитель $(m - 1)$ отвечает за количество рассмотренных вариантов разбиения вершины на две, из которых в итоге выбирается только один (поэтому -1). В представленных далее примерах выполняется сравнение точности и производительности представленного алгоритма с методом Монте-Карло, для которого критерий (1.7) тождественно равен количеству симуляций.

В процессе дробления сеток погрешность интерполяции изменяется скачкообразно, и существует некоторое значение $\tau > 0$, такое, что перемещение решения в момент $t_k + \tau$ не приведет к дроблению сеток, и, следовательно, функция $N_i(t_k) + (m-1)N_{new}(t_k)$ является кусочно-постоянной. Отсюда вытекает существование такого числа N и соответствующего разбиения $\{t_k\}_{k=\overline{0,N}}$, при которых достигается точное значение критерия (1.7). На практике число N ограничено сверху вследствие того, что в контексте производительности величина $t_k - t_{k-1}$ не может быть меньше шага интегрирования. Исходя из вышеизложенного, можно привести рекомендации по выбору шага дискретизации τ: если на текущем шаге дроблений вершин не было, то шаг увеличивается, в противном случае — уменьшается.

Утверждение 2 является очень важным результатом с научной и практической точки зрения. Оно означает, что оценка глобальной погрешности не зависит от общего числа шагов по времени. И выбор разбиения $\{t_k\}_{k=\overline{0,N}}$ на нее практически не влияет. Также высота kd-дерева является логарифмической функцией и растет очень медленно. К примеру, если предположить, что десятимерную область неопределенности необходимо по каждому измерению разделить на 1000, то есть в сумме на 1000^{10} частей, то высота kd-дерева в этом случае не будет превышать 100. К слову, число 10^{30} колоссальное и превышает даже объем хранимых данных на Земле. Отметим: так как kd-дерево в процессе работы алгоритма не только растет в высоту, но и уменьшается, то оценка глобальной погрешности тоже может не только возрастать, но и уменьшаться.

1.8. РЕЗУЛЬТАТЫ

Все расчеты выполнены с позиции минимизации вычислительных затрат, то есть с использованием равномерных сеток и дроблением вершин всегда пополам, а также с использованием второго подхода к оценке погрешности в вершинах, основанного на разрежении интерполяционной сетки. Степень интерполяционного многочлена равна четырем. Относительная погрешность — 10^{-5} (1.5). Для оценки глобальной погрешности в полученное решение подставляются точечные значения интервальных параметров и выполняется сравнение с решениями соответствующих неинтервальных систем ОДУ.

Вначале рассмотрим систему ОДУ, содержащую в себе одно интервальное начальное условие:

$$\begin{cases} x' = -\dfrac{y}{\sqrt{x^2 + y^2}}, \\ y' = \dfrac{x}{\sqrt{x^2 + y^2}}, \\ x(0) = x_0 \in [1, 9],\ y(0) = 0, \\ t \in [0, 100]. \end{cases} \tag{1.8}$$

Ее аналитическое решение имеет вид:

$$x(x_0,t) = x_0 \cos\left(\frac{t}{x_0}\right),$$
$$y(x_0,t) = x_0 \sin\left(\frac{t}{x_0}\right). \qquad (1.9)$$

Для нахождения точной интервальной оценки решения (1.9) в конечный момент времени $t_N = 100$ необходимо решить несколько задач на поиск экстремума:

$$\mathbf{x}(t_N) = \left[\min_{x_0\in[0,\,9]} x(x_0,t_N),\ \max_{x_0\in[0,\,9]} x(x_0,t_N)\right],$$
$$\mathbf{y}(t_N) = \left[\min_{x_0\in[0,\,9]} y(x_0,t_N),\ \max_{x_0\in[0,\,9]} y(x_0,t_N)\right].$$

В табл. 1.1 приведены результаты расчетов. Для метода Монте-Карло значение критерия I (1.7) равняется количеству симуляций. С учетом дополнительных временных затрат, связанных с работой над kd-деревом, предлагаемый алгоритм работает в 20–30 раз быстрее.

Таблица 1.1

Сравнение результатов решения системы (1.8) различными методами

Метод	$\mathbf{x}(t_N)$	$\mathbf{y}(t_N)$	I
Точное решение	[–6.379155, 7.983118]	[–8.939997, 7.091346]	–
Алгоритм адаптивной интерполяции	[**–6.37915**8, **7.98312**2]	[**–8.939997**, **7.09134**5]	354
Монте-Карло	[**–6.37915**2, **7.98311**6]	[**–8.93**8380, **7.09134**5]	10000

На рис. 8 показано множество решений системы (1.8) на фазовой плоскости. В начальный момент времени $t_0 = 0$ оно представляет собой отрезок, параллельный оси абсцисс. В процессе интегрирования системы множество растягивается и закручивается в спираль.

На рис. 9 отображена зависимость решений от интервального параметра. Здесь наблюдаются периодические колебания с увеличением амплитуды. На рис. 10 представлены интервальные оценки решений от времени. Так как вклад в интервальную оценку, в основном, вносят решения с начальным условием, близким к $x_0 = 9$, то присутствующие здесь колебания соответствуют колебаниям правого конца кривых, представленных на рис. 9.

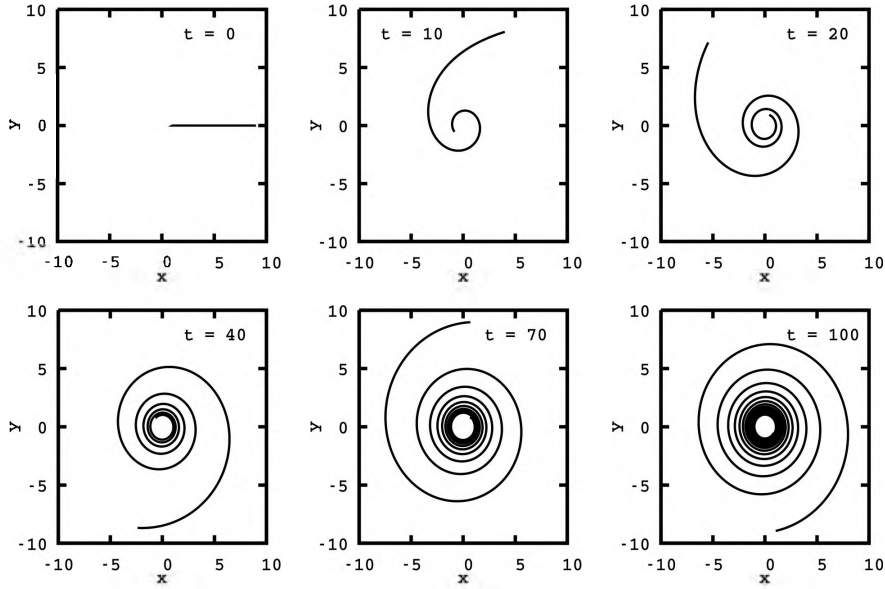

Рис. 8. Множество решений системы (1.8) в различные моменты времени

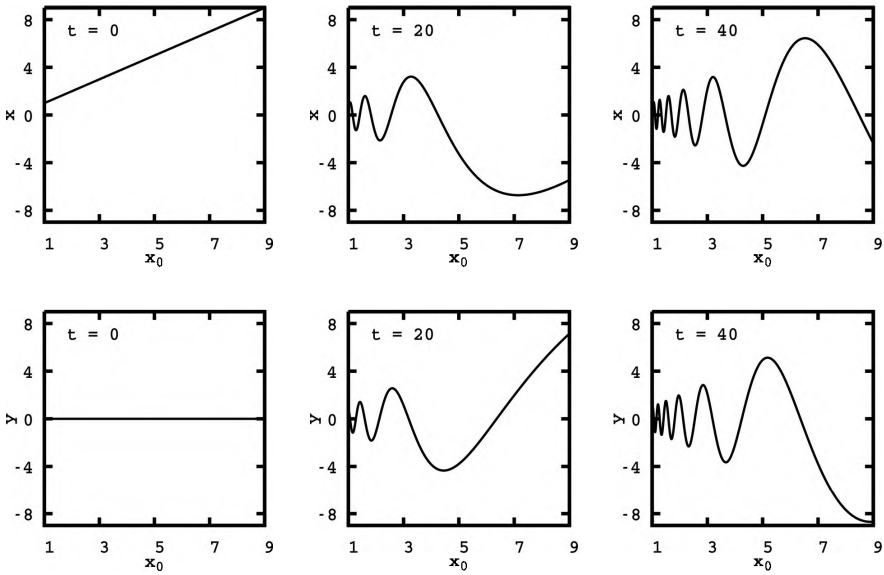

Рис. 9. Зависимость решения системы (1.8) от интервального параметра

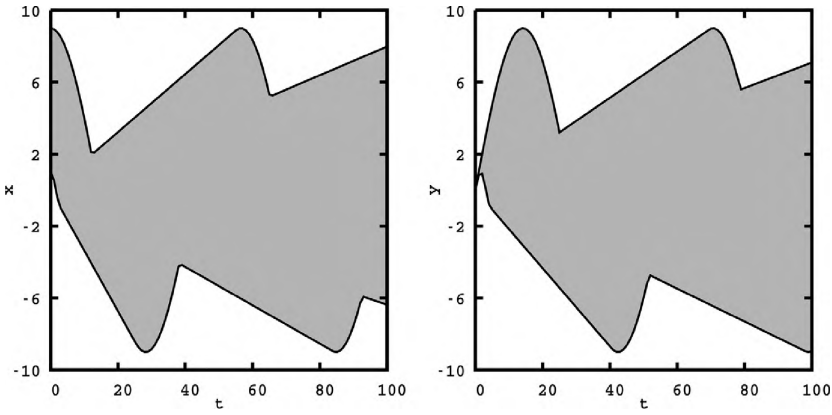

Рис. 10. Зависимость верхних и нижних оценок решения системы (1.8) от времени

На рис. 11 показана разница между полученным решением и точным. Увеличение ошибки ближе к левому краю графика объясняется уплотнением периодов колебаний решений (см. рис. 9), что приводит к сгущению сетки, росту kd-дерева (рис. 12) и увеличению погрешности (утверждение 2).

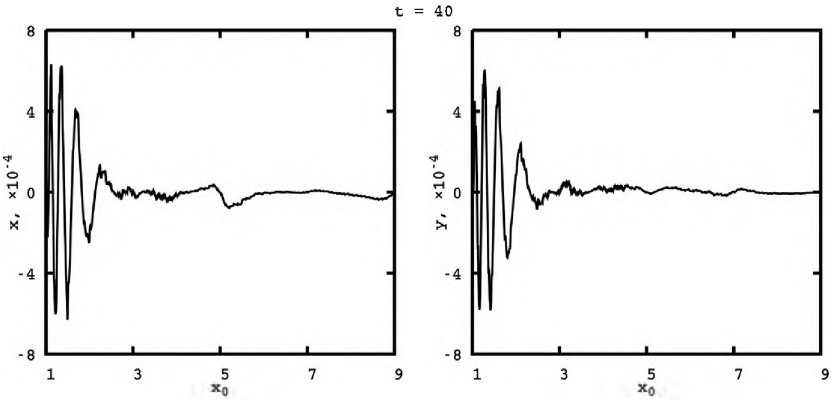

Рис. 11. Разница между аналитическим (1.9) и численным решениями

В рассматриваемых ниже системах ОДУ нахождение аналитического решения затруднительно. Поэтому в качестве точного решения в конечный момент времени t_N используется решение, полученное

классическими методами оптимизации с заведомо большой точностью (в приведенных расчетах использовался алгоритм роя частиц — Inertia Weighted PSO).

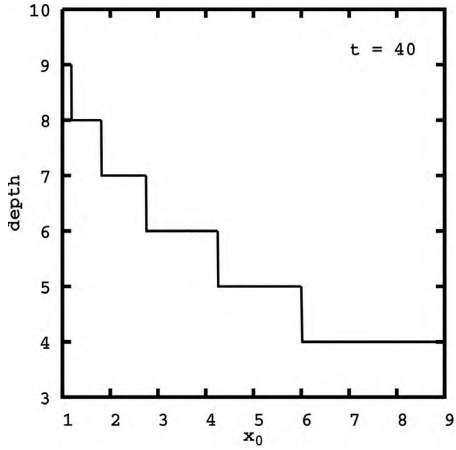

Рис. 12. Глубина вершины, в которой содержится решение для конкретного значения интервального параметра в kd-дереве

Далее рассмотрим систему ОДУ с двумя интервальными начальными условиями, которая описывает консервативный осциллятор:

$$\begin{cases} x' = y, \\ y' = -\sin(x), \\ x(0) \in [-1, 1],\ y(0) \in [0, 1], \\ t \in [0, 40]. \end{cases} \tag{1.10}$$

Как видно из табл. 1.2, один миллион симуляций в методе Монте-Карло гарантирует только 2–4 знака в решении. Алгоритм адаптивной интерполяции получает решение намного быстрее и на порядки точнее.

Таблица 1.2

Сравнение результатов решения системы (1.10) различными методами

Метод	$\mathbf{x}(t_N)$	$\mathbf{y}(t_N)$	I
Точное решение	[–1.378438, 1.341918]	[–1.289731, 1.099147]	–
Алгоритм адаптивной интерполяции	[**–1.378**44, **1.341918**]	[**-1.289731**, **1.09914**4]	1579
Монте-Карло	[**–1.378**185, **1.3**30360]	[**–1.28**7727, **1.099**029]	100000

На рис. 13 показаны зависимости интервальных оценок решений от времени. Здесь четко видны изломы на графиках. Это связано с тем, что точка, соответствующая верхней или нижней границе на множестве решений, в определенный момент перескакивает с одного места на другое и ее траектория является разрывной.

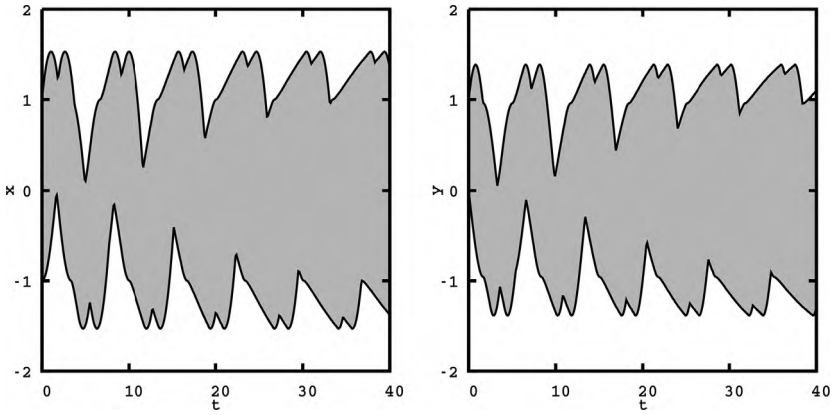

Рис. 13. Зависимость верхних и нижних оценок решения системы (1.10) от времени

В процессе интегрирования множество решений закручивается в спиралевидную структуру (рис. 14). В определенный момент оно напоминает кошку, далее — привидение, а в конце — знаки инь и ян. На рис. 15 представлены получающиеся в процессе работы алгоритма разбиения пространства. Линии на этих двух рисунках соответствуют друг другу.

На рис. 16 отображено множество решений в форме поверхности в трехмерном пространстве. На верхних графиках дополнительному измерению соответствует либо один, либо второй интервальный параметр. На нижних графиках представлена зависимость каждого решения в отдельности от x_0 и y_0.

На рис. 17 наблюдается корреляция глобальной погрешности для конкретных начальных условий и глубин соответствующих вершин в kd-дереве.

Далее представлены задачи, содержащие не только интервальные начальные условия, но и интервальные параметры. При выполнении

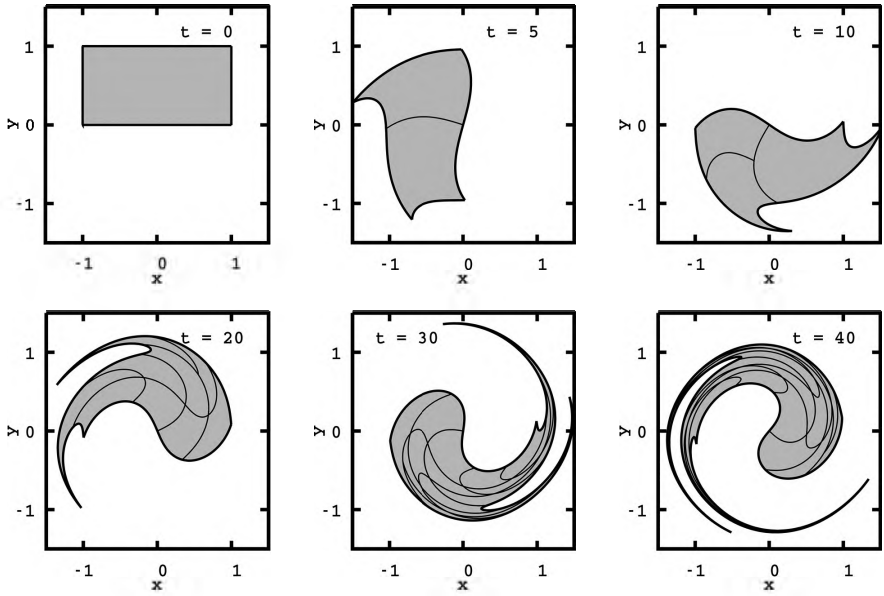

Рис. 14. Множество решений системы (1.10) в различные моменты времени

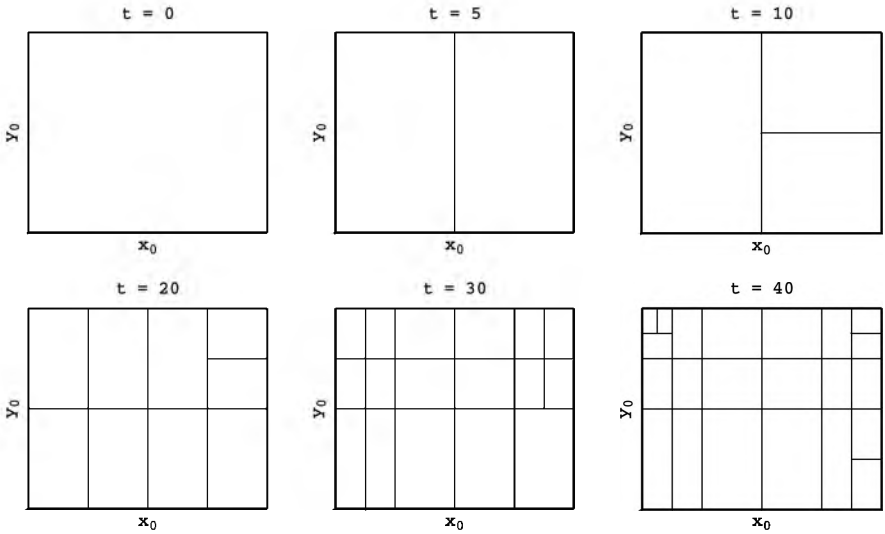

Рис. 15. Разбиения пространства, получающиеся в процессе решения системы (1.10)

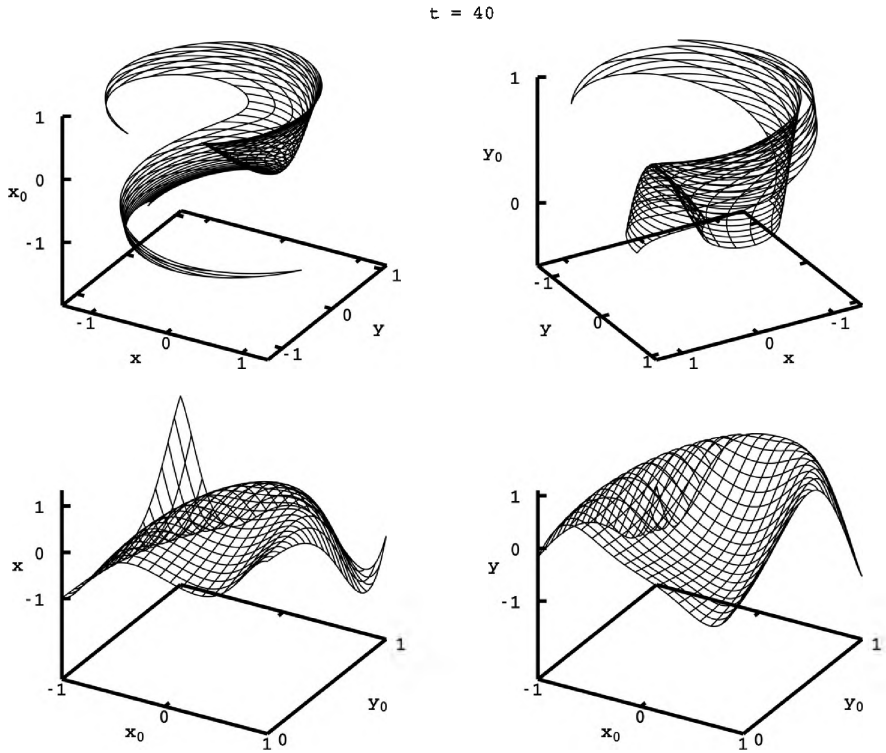

Рис. 16. Зависимость решения системы (1.10) от интервальных параметров

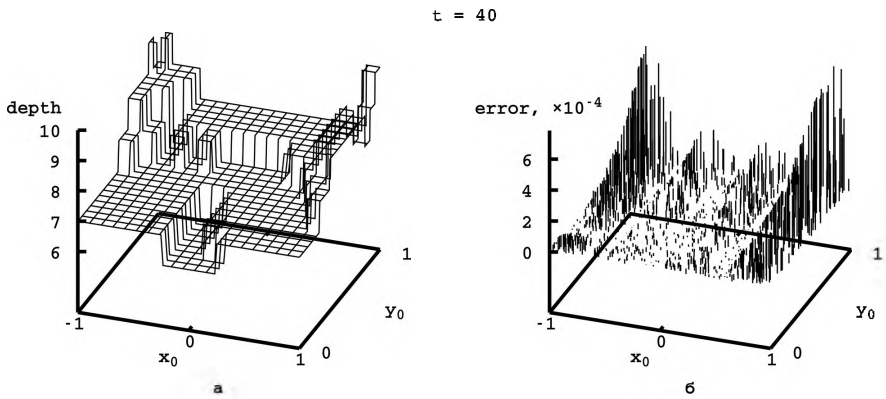

Рис. 17. Распределение глобальной погрешности (*б*) и глубин соответствующих вершин (*а*) в kd-дереве для системы (1.10)

вычислений в систему ОДУ добавлялись фиктивные уравнения, для того чтобы интервальные данные содержались только в начальных условиях.

Рассмотрим систему ОДУ, соответствующую модели Лотки–Вольтерры [81]:

$$\begin{cases} x' = -0.9x + 0.5xy, \\ y' = \alpha y - 0.8xy, \\ x(0) \in [0.9, 1.1],\ y(0) \in [1.9, 2.1], \\ t \in [0, 100], \end{cases} \tag{1.11}$$

где $\alpha \in [0.7, 0.75]$.

В табл. 1.3 приведено сравнение решений, полученных разными методами. Сто миллионов симуляций методом Монте-Карло гарантируют только 3 знака в решении, в то время как алгоритм адаптивной интерполяции при существенно меньших затратах дает решение с точностью до 6-го знака. На рис. 18 показана проекция множества решений из трехмерного пространства (см. рис. 19) в двухмерное пространство фазовых переменных. Дополнительному измерению соответствует интервальный параметр α. Сначала множество решений является прямоугольным параллелепипедом, который в процессе решения претерпевает сильные деформации. На рис. 20 показано разбиение пространства в различные моменты времени.

Таблица 1.3

Сравнение результатов решения системы (1.11) различными методами

Метод	$\mathbf{x}(t_N)$	$\mathbf{y}(t_N)$	I
Точное решение	[0.6502434, 1.126302]	[1.615649, 2.248024]	–
Алгоритм адаптивной интерполяции	[**0.6502**446, **1.12630**1]	[**1.615649**, **2.248024**]	387
Монте-Карло	[**0.650**3192, **1.126**165]	[**1.61**6225, **2.24**7694]	100000000

На рис. 21 отображено множество решений в пространствах $x \times y \times x_0$ и $x \times y \times y_0$. В этих пространствах оно выглядит как полукруглая «стенка», которая вращается и со временем растягивается. На рис. 22 представлены интервальные оценки решения. Так как множество решений с течением времени увеличивается в размерах, то, соответственно, и интервальный «коридор» со временем становится более широким.

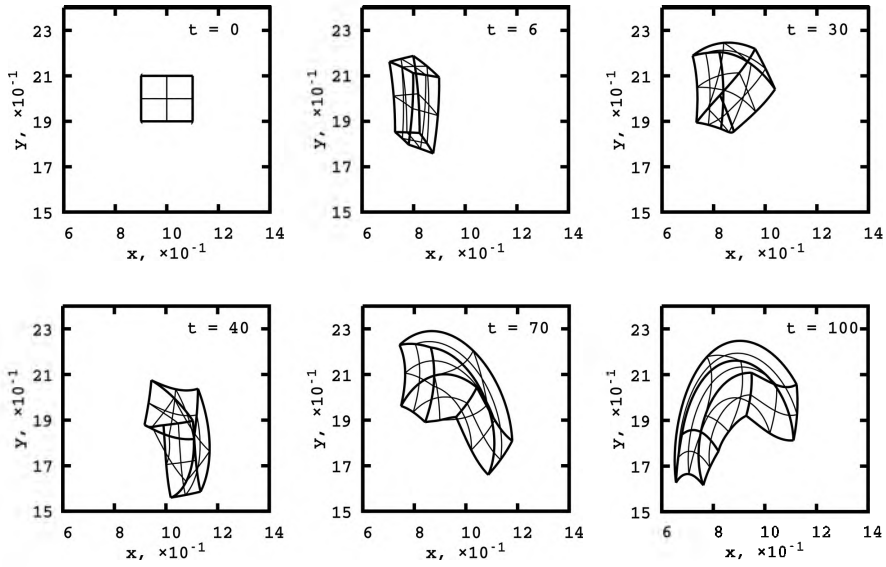

Рис. 18. Множество решений системы (1.11) в различные моменты времени

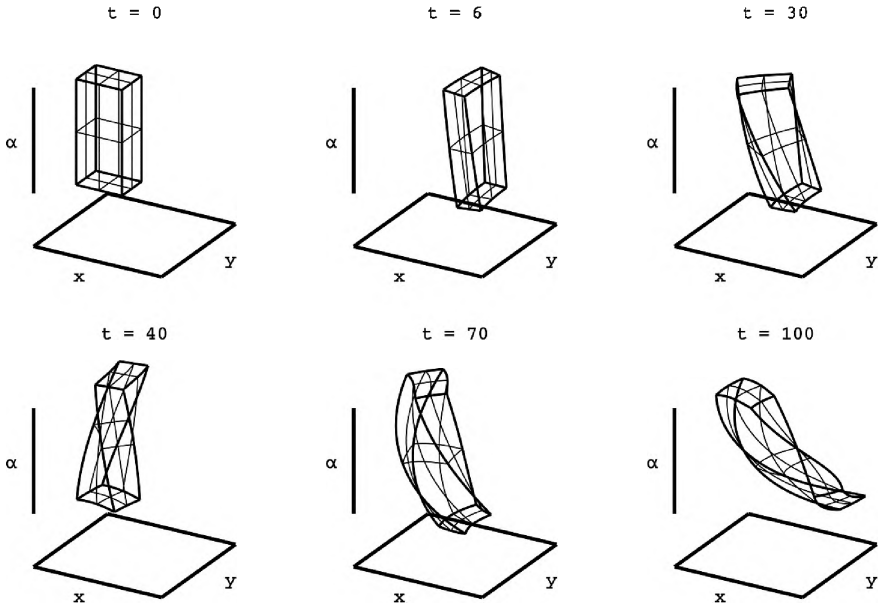

Рис. 19. Множество решений системы (1.11) в трехмерном пространстве

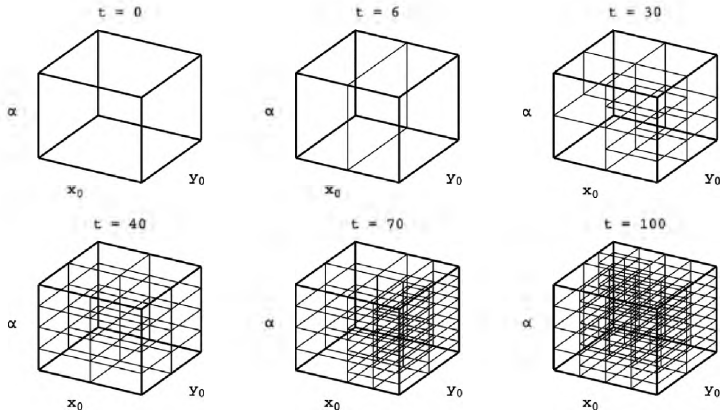

Рис. 20. Разбиения пространства, получающиеся в процессе решения системы (1.11)

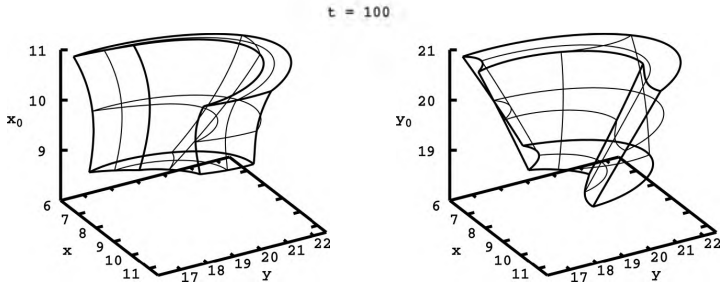

Рис. 21. Зависимость решения системы (1.11) от интервальных параметров

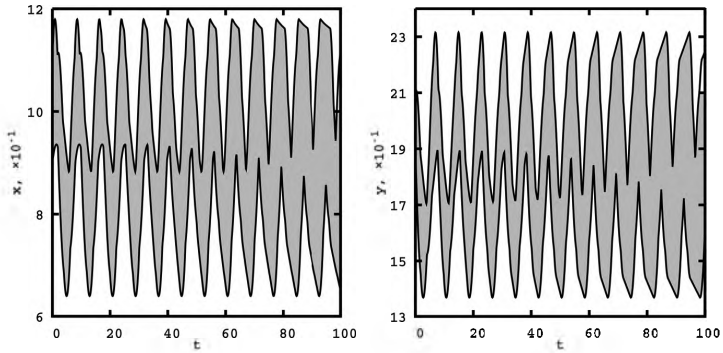

Рис. 22. Зависимость верхних и нижних оценок решения системы (1.11) от времени

Следующая система ОДУ описывает взаимодействие двух тел под действием сил гравитации с интервальными начальными положениями тел и интервальными массами:

$$\begin{cases} x_1' = v_1^x,\ y_1' = v_1^y,\ x_2' = v_2^x,\ y_2' = v_2^y, \\ \left(v_1^x\right)' = m_2 \dfrac{x_2 - x_1}{r^3},\ \left(v_1^y\right)' = m_2 \dfrac{y_2 - y_1}{r^3}, \\ \left(v_2^x\right)' = m_1 \dfrac{x_1 - x_2}{r^3},\ \left(v_2^y\right)' = m_1 \dfrac{y_1 - y_2}{r^3}, \\ x_1(0) \in [-1.005,\ 1.005],\ y_1(0) = 0, \\ v_1^x(0) = 0,\ v_1^y(0) = 0.4, \\ x_2(0) = 1,\ y_2(0) \in [-0.005,\ 0.005], \\ v_2^x(0) = 0,\ v_2^y(0) = -0.4, \\ t \in [0,\ 100], \end{cases} \tag{1.12}$$

где $r = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, $m_1 \in [0.995,\ 1.005]$, $m_2 \in [0.995,\ 1.005]$.

В табл. 1.4 приведено сравнение результатов решения системы различными методами. Метод Монте-Карло дает только 1–2 знака в решении, причем для получения результата потребовалось несколько дней расчетов при условии его распараллеливания на графическом процессоре, содержащем более 1000 cuda-ядер.

Таблица 1.4

Сравнение результатов решения системы (1.12) различными методами

t_N	Точное решение	Алгоритм адаптивной интерполяции	Монте-Карло
$\mathbf{x}_1$	[–0.5748170, 0.4758124]	[**–0.5748170**, **0.475812**5]	[**–0.57**35955, **0.475**7944]
$\mathbf{y}_1$	[–0.8870650, 0.1647035]	[**–0.8870650**, **0.1647035**]	[**–0.88**65736, **0.16**18549]
$\mathbf{v}_1^x$	[–0.6300935, 0.2122600]	[**–0.630093**4, **0.2122600**]	[**–0.6300**659, **0.2**075954]
$\mathbf{v}_1^y$	[–0.8448854, 0.2008404]	[**–0.844885**3, **0.2008404**]	[**–0.8448**288, **0.200**2040]
$\mathbf{x}_2$	[–0.4808124, 0.5798170]	[**–0.480812**5, **0.5798170**]	[**–0.480**7869, **0.57**85901]
$\mathbf{y}_2$	[–0.1697035, 0.8907500]	[**–0.1697035**, **0.890749**9]	[**–0.16**66990, **0.8**898884]
$\mathbf{v}_2^x$	[–0.2122600, 0.6300935]	[**–0.2122600**, **0.630093**4]	[**–0.2**076016, **0.6300**812]
$\mathbf{v}_2^y$	[–0.2008404, 0.8448854]	[**–0.2008404**, **0.8448854**]	[**–0.200**2052, **0.8448**734]
I	–	26497	231000000

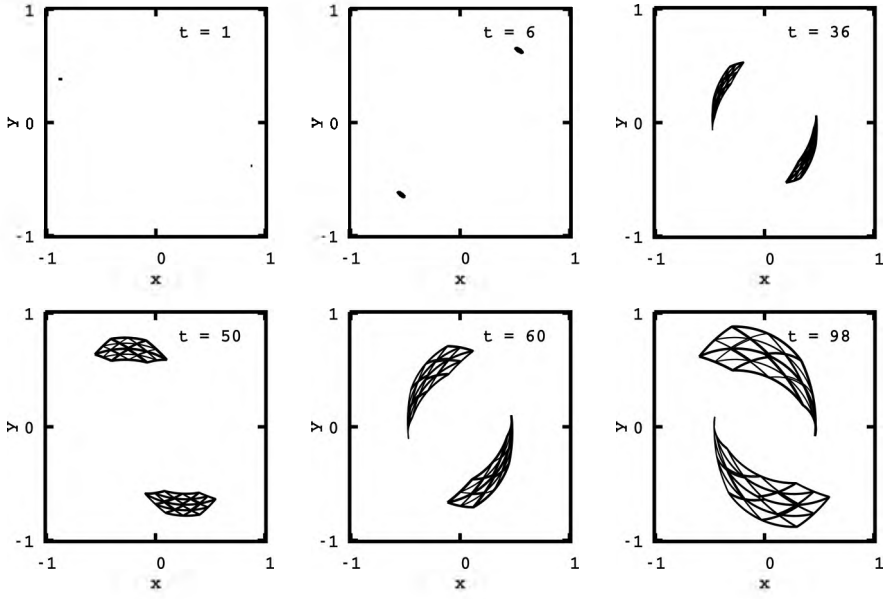

Рис. 23. Положения тел в пространстве в различные моменты времени

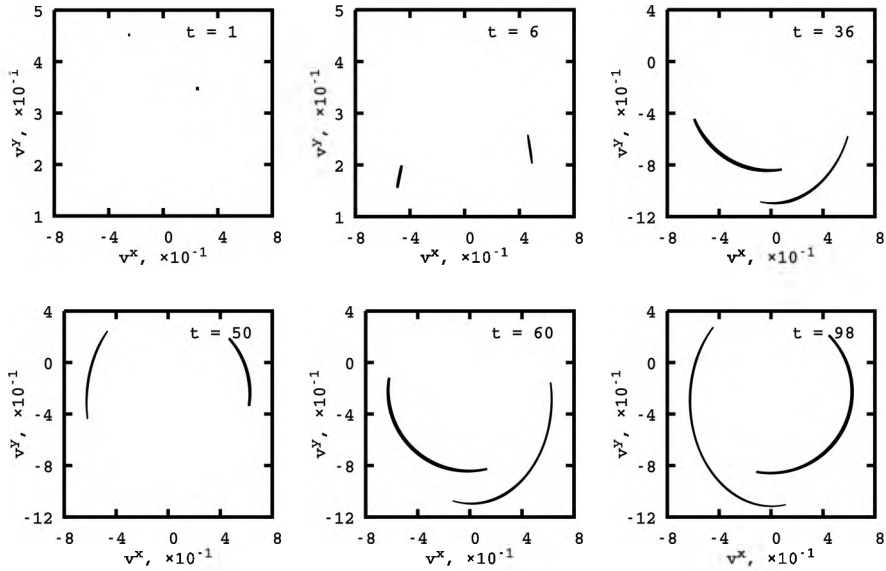

Рис. 24. Скорости тел в различные моменты времени

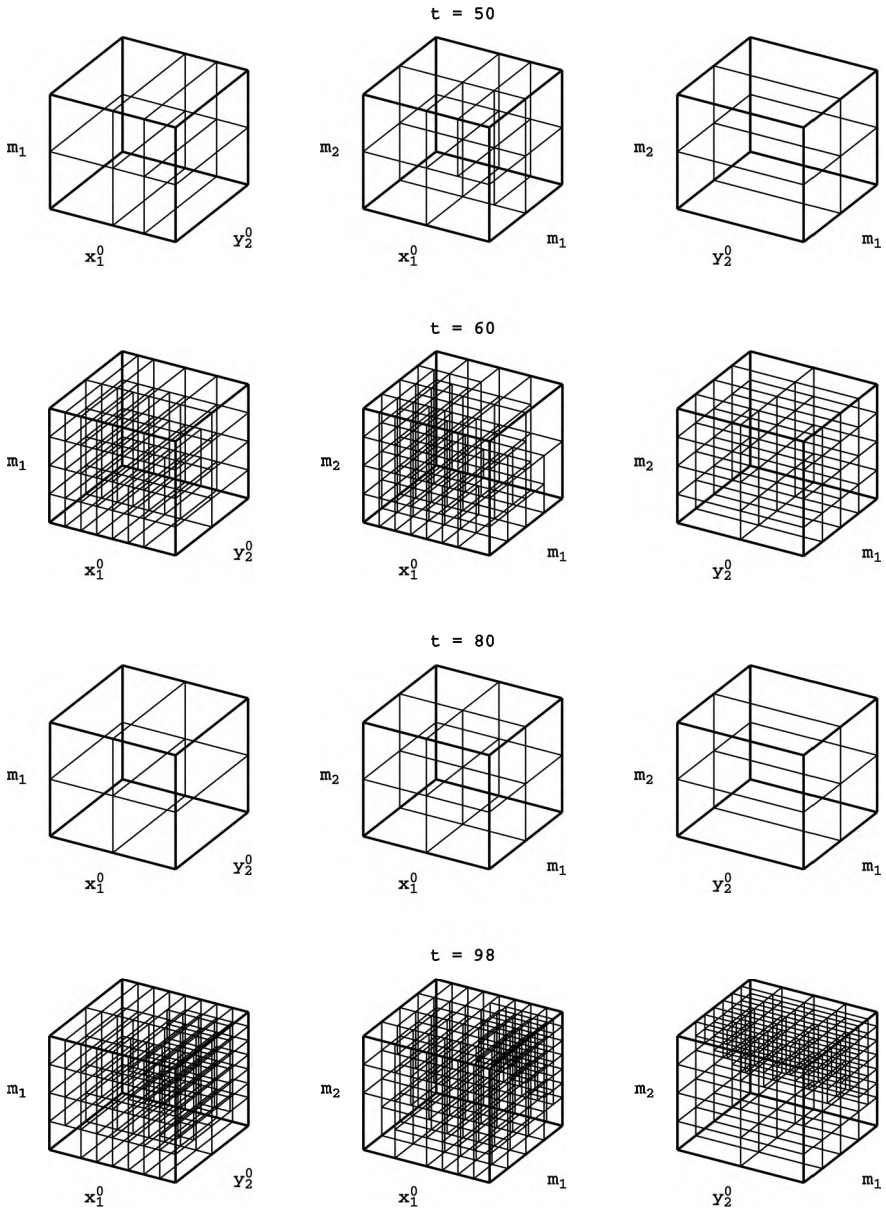

Рис. 25. Некоторые проекции разбиения пространства, получающиеся в процессе решения системы (1.12)

На рис. 23 и 24 показаны два множества решений, соответствующие скорости и положению каждого тела в пространстве. При этом между ними существует биекция, то есть для каждой точки из первого множества существует только одна из второго, и наоборот. На рис. 23 изображена проекция четырехмерного множества, находящегося в десятимерном пространстве, на координатную плоскость. Каждое множество представляет собой проекцию деформирующегося гиперкуба.

В этом примере сетка строится в четырехмерном пространстве, и, чтобы ее проиллюстрировать, на рис. 25 приведены несколько ее проекций на различные трехмерные гиперплоскости. Движения тел являются периодическими: они то сближаются, то отдаляются друг от друга. При этом на этапе сближения происходит наибольшее влияние тел друг на друга, и в этот момент множества их положений претерпевают сильные деформации, что отражается на сгущении сетки. А в момент, когда тела максимально удалены друг от друга, происходит уменьшение ее плотности. Эта периодичность также отражается на высоте kd-дерева и глобальной погрешности (рис. 26). На рис. 27 показаны интервальные оценки решений в зависимости от времени. Здесь тоже наблюдаются периодичность (как на рис. 22) и увеличение со временем ширины интервальных оценок.

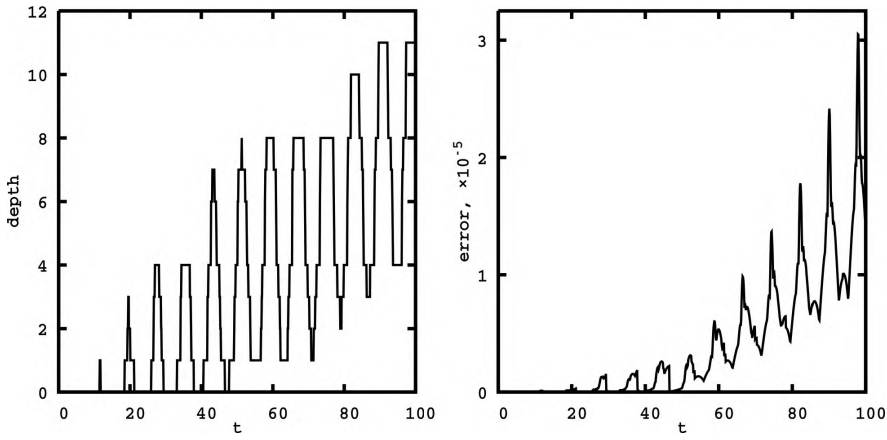

Рис. 26. Зависимость высоты kd-дерева (слева) и величины глобальной погрешности (справа) от времени в процессе интегрирования системы (1.12)

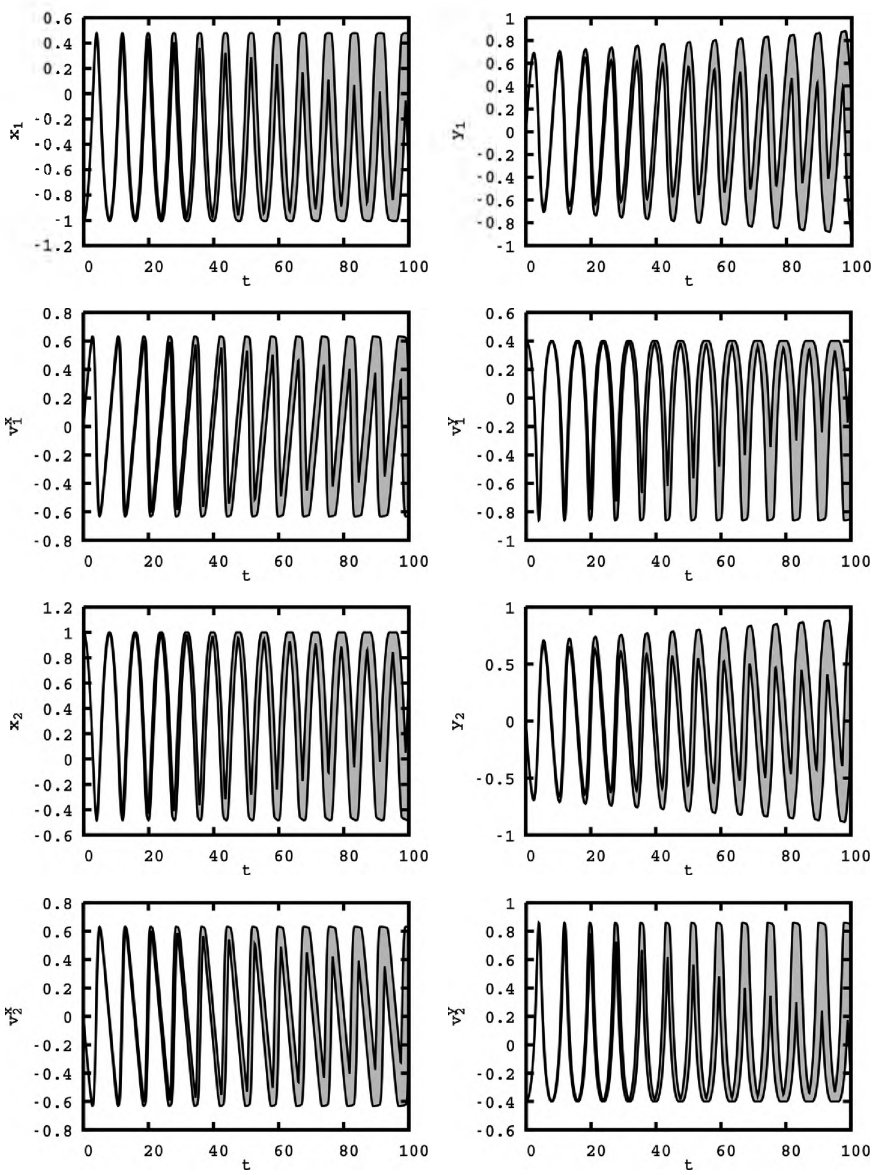

Рис. 27. Зависимость верхних и нижних оценок решения системы (1.12) от времени

1.9. ЗАКЛЮЧЕНИЕ

Рассмотрен алгоритм адаптивной интерполяции для задач моделирования динамических систем с интервальными параметрами, который позволяет за приемлемое время находить интервальную оценку решений с контролируемой точностью, не подвержен эффекту обертывания, имеет высокую степень распараллеливания и справляется с «большими» интервалами. Алгоритм заключается в построении над множеством, образованным интервальными параметрами, динамической структурированной сетки (адаптивного разбиения пространства) на основе kd-дерева. Сформулированы и доказаны утверждения, касающиеся условий применимости алгоритма и его глобальной погрешности. Показано, что оценка глобальной погрешности прямо пропорциональна высоте kd-дерева. Тестирование алгоритма на представительном ряде примеров разной размерности, содержащих различное количество интервальных параметров, демонстрирует его эффективность, а также превосходство над методом Монте-Карло по вычислительным затратам.

Важно отметить тот факт, что если количество интервальных начальных условий совпадает с количеством уравнений в системе, то достаточно рассматривать только границу области неопределенности, это позволяет уменьшить вычислительную сложность решаемой задачи. Однако на практике такая ситуация встречается редко.

ГЛАВА 2. ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ МОДЕЛИРОВАНИЯ ДИНАМИЧЕСКИХ СИСТЕМ С ИНТЕРВАЛЬНЫМИ ПАРАМЕТРАМИ

2.1. ТЕХНОЛОГИЯ CUDA

В последние годы активно идет развитие программно-аппаратной технологии CUDA, которая позволяет использовать графические процессоры (GPU) компании NVIDIA для общих вычислений [82, 83]. Ключевое отличие GPU от центральных процессоров (CPU) заключается в наличии тысяч ядер, способных одновременно производить расчеты. Зачастую при использовании даже не очень дорогих видеокарт можно получить прирост производительности в десятки, а то и в сотни раз по сравнению с вычислениями на центральном процессоре. Отметим, что при всей своей привлекательности разработка, ориентированная на GPU, обладает рядом особенностей. Например, вместо наличия только оперативной памяти здесь имеется шесть различных ее видов и возможность напрямую взаимодействовать с кеш-памятью.

Далее приводится сравнение устройств центрального процессора и графического, а также описание программной части технологии CUDA из работы [71].

На рис. 28 схематически изображена архитектура CPU. Вычислительные модули (АЛУ) здесь занимают существенно меньшую часть кристалла по сравнению с различными видами кеш-памяти. Каждый из АЛУ представляет собой полноценный процессор и поддерживает все соответствующие функции. Такая архитектура подходит для нормальной работы обычного программного обеспечения, которым пользуется большинство пользователей персональных компьютеров (ПК). Но при использовании центрального процессора как векторного вычислительного модуля, который должен выполнять много одинаковых операций над разными данными, существенная часть его возможностей становится излишней и ненужной. Потоки, которые поддерживаются, являются «тяжеловесными» и управляются операционной системой, и, следовательно, их не может быть много.

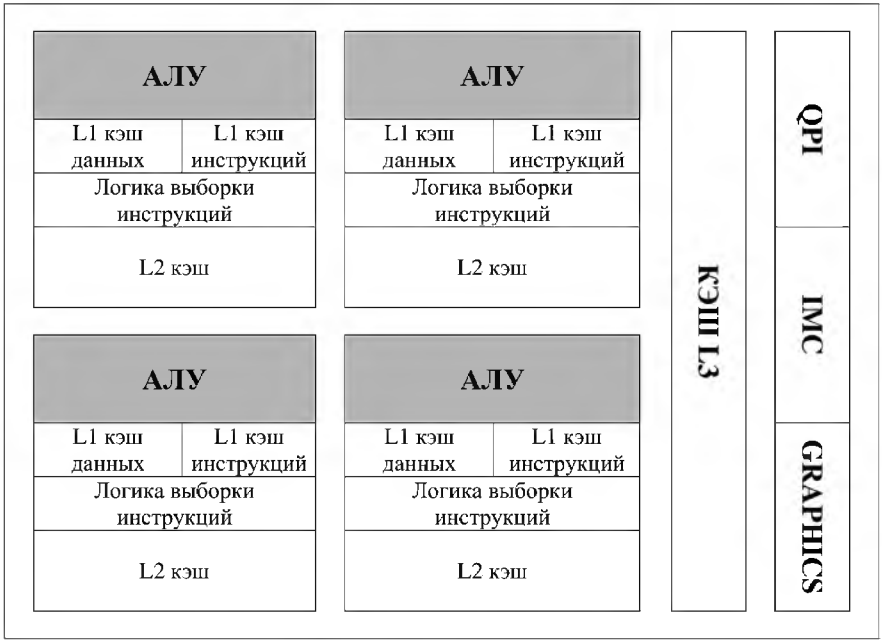

Рис. 28. Архитектура CPU

На рис. 29 схематично представлена архитектура графического процессора. Здесь существенную часть кристалла занимают вычислительные устройства (ядра). Ядра GPU устроены проще, чем ядра CPU, их основная задача — выполнять только математические операции. Они несамостоятельные и являются сопроцессорами к центральному процессору. Количество ядер в современных видеокартах измеряется тысячами, а суммарная их производительность — десятками терафлопсов. Такое устройство GPU объясняется тем, что при решении задач рендеринга изображения необходимо выполнять большое количество однотипных независимых математических вычислений.

На аппаратном уровне все вычислительные ядра объединены в кластеры текстурных процессоров (TPC). Каждый кластер состоит из укрупненного блока текстурных выборок и нескольких потоковых мультипроцессоров (SM). Каждый мультипроцессор состоит из определенного количества потоковых процессоров (SP), блоков SFU (Super Function Unit) для расчета «тяжелых» математических функций (экс-

понента, синус, косинус, корень и т.д.), различных видов кеш-памяти и т.д. (рис. 30). На уровне одного SM все инструкции выполняются по принципу SIMD (одна инструкция — много данных).

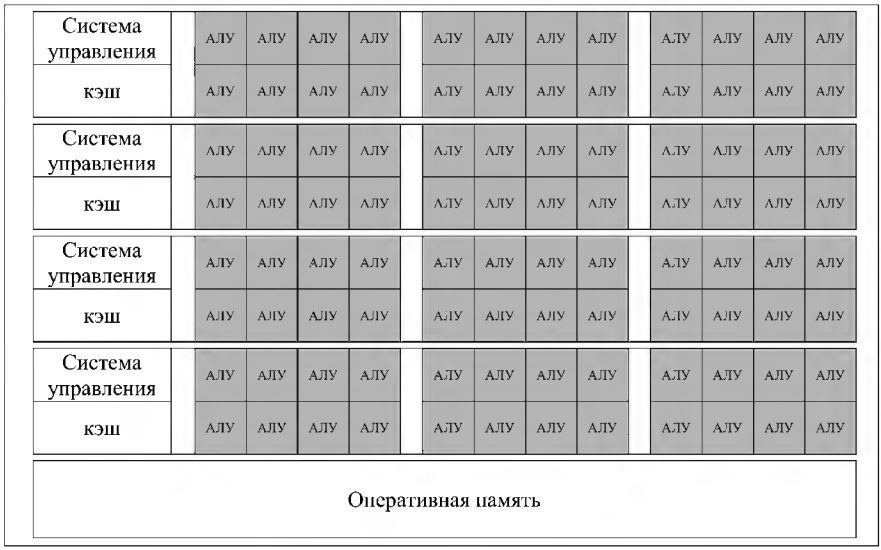

Рис. 29. Архитектура GPU

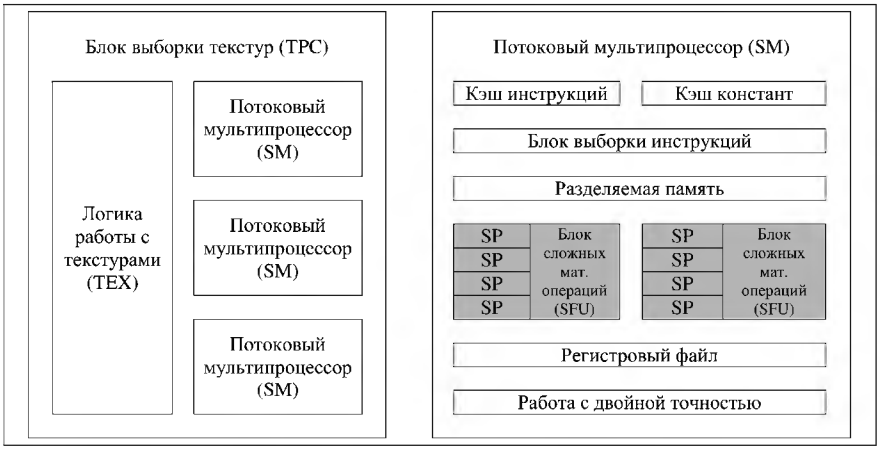

Рис. 30. Устройство TPC и SM

Ниже описывается программная часть технологии CUDA (рис. 31).

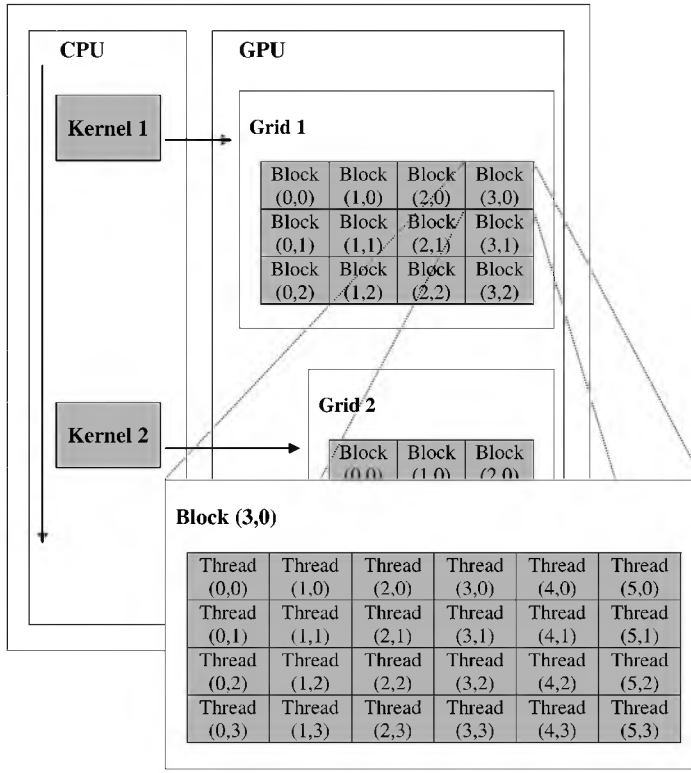

Рис. 31. Ядра, сетки, блоки и потоки

Основные понятия:

- Устройство (Device) — графический процессор, видеокарта.
- Хост (Host) — центральный процессор.
- Ядро (Kernel) — специальная функция в программе, которая запускается центральным процессором и выполняется на устройстве.
- Тред (Thread) — элементарный поток исполнения на устройстве.
- Блок (Block) — объединение потоков. Все потоки одного блока выполняются на одном мультипроцессоре (SM) и могут иметь общую память.

- Грид (Grid) — объединение блоков. Все блоки одного грида выполняются на одном устройстве.
- Варп (Warp) — группа из 32 потоков, которые выполняются физически одновременно. Элементарная единица выполнения.

Центральный процессор в основном занимается подготовительной работой, а именно: размещает данные в памяти GPU, выполняет конфигурацию и запуск ядра. Вызов ядра в общем случае является асинхронным, то есть после его вызова поток исполнения на CPU не зависает, а идет дальше.

Всего существует шесть видов памяти:

- Регистровая память. Располагается на мультипроцессоре, используется для хранения локальных переменных. Размер ограничен (32 Кб). Скорость доступа самая быстрая. Доступ имеется на уровне одного треда.
- Локальная память. Используется в случае нехватки регистров. Скорость доступа низкая, из-за того что она расположена в DRAM. Выделяется отдельно для каждого треда. Неуправляемая, компилятор самостоятельно решает, какие переменные разместить в регистровой памяти, а какие — в локальной. Ситуация, когда начинает задействоваться данный тип памяти, называется «спиллинг регистров».
- Разделяемая память. Располагается на мультипроцессоре. Скорость доступа высокая, но чуть меньше, чем у регистровой памяти. Размер ограничен (16–48 Кб). Общая для всех потоков одного блока. Выделяется на один блок. Разделена на 32 банка памяти. Обращения на уровне одного банка выполняются последовательно.
- Глобальная память. Основная память видеокарты. Расположена в микросхемах DRAM. Имеет низкую скорость доступа. Выделяется на уровне всей программы. Для эффективной работы необходимо, чтобы все потоки одного варпа обращались к непрерывному участку памяти.
- Константная память. Располагается в DRAM. Кешируется. Размер ограничен (65 Кб). Выделяется на уровне всей программы. Задействуется компилятором для размещения в ней аргументов вызова ядра.

– Текстурная память. Располагается в DRAM. Использует специальный текстурный кеш. Выделяется на уровне всей программы.

Ядро представляет собой функцию в программе, помеченную специальным модификатором `__global__`:

```
__global__ void kernel(args) {
     int idx = blockIdx.x * blockDim.x + threadIdx.x;
     int offset = blockDim.x * gridDim.x;
     ...
}
```

Внутри любой функции, работающей на GPU, то есть вызванной из ядра, доступны несколько особых переменных:

– `dim3 gridDim` — размер грида;
– `uint3 blockIdx` — координаты текущего блока внутри грида;
– `dim3 blockDim` — размер блока;
– `uint3 threadIdx` — координаты текущего треда внутри блока,

где `dim3` и `uint3` — новые типы данных (по сравнению с типами данных языка Си), представляющие собой векторы из трех компонентов.

Запуск ядра осуществляется следующим образом:

```
kernel<<<grid, block, memory, stream>>>(args);
```

где:

– `dim3 grid` — размер грида;
– `dim3 block` — размер блока;
– `size_t memory` — количество разделяемой памяти на блок в байтах;
– `cudaStream_t stream` — описание потока, в котором запускается ядро.

Последние два аргумента являются необязательными.

В целом разработка программного обеспечения под графические процессоры обладает рядом особенностей. Аспекты, о которых можно не задумываться при разработке под центральные процессоры, здесь требуют особого внимания. Зачастую приходится жертвовать «кра-

сотой» программного кода в угоду повышению производительности, экономить локальные переменные, частично отказаться от объектно-ориентированного программирования и т.д.

2.2. ПОСТАНОВКА ЗАДАЧИ МОДЕЛИРОВАНИЯ НА ГРАФИЧЕСКИХ ПРОЦЕССОРАХ

Процесс моделирования динамической системы, содержащей m интервальных параметров, можно представить в виде последовательности отображений:

$$\begin{cases} y_{k+1} = F_k(y_k), \\ y_0 \in \mathbf{y_0} = \left(\mathbf{y_0^1}, \mathbf{y_0^2}, ..., \mathbf{y_0^m}, y_0^{m+1}, ..., y_0^n\right)^T, \\ k = 0, ..., N-1, \end{cases}$$

где $F_k : \mathbb{R}^n \to \mathbb{R}^n$, $\mathbf{y_0} \in \mathbb{IR}^n$ — интервальные начальные условия. Вектор-функция F_k удовлетворяет условиям, обеспечивающим существование и единственность решения при всех $y_0 \in \mathbf{y_0}$.

Если динамическая система задается с помощью системы ОДУ, тогда $F_k(y_k) = y(t_{k+1})$ и представляет собой решение соответствующей системы ОДУ:

$$\begin{cases} y' = f(y), \\ y(t_k) = y_k, \\ t \in [t_k, t_{k+1}]. \end{cases}$$

В случае дискретной динамической системы F_k описывает закон перехода из одного состояния в другое.

2.3. РАСПАРАЛЛЕЛИВАНИЕ И РЕАЛИЗАЦИЯ

Ниже приведен псевдокод описанного в первой главе алгоритма. В цикле по переменной k выделяются три блока, каждому из которых соответствует свой вложенный цикл. Первый блок переносит решения, содержащиеся в узлах интерполяционных сеток, на следующий временной слой. Второй блок выполняет удаление вершин из дере-

ва, для которых погрешность интерполяции является приемлемой. Третий блок отвечает за разбиение вершин.

В представленном псевдокоде функция childs возвращает список всех потомков; is_leaf проверяет, является ли вершина листовой; eval_error вычисляет погрешность интерполяции; build_split разбивает вершину на две.

Если рассмотреть алгоритм с точки зрения распараллеливания, то один шаг алгоритма разбивается на следующие этапы:

1) применение ко всем узлами сеток отображения F_k;

2) вычисление погрешности интерполяции во всех вершинах дерева;

3) выполнение удаления и разбиения вершин.

Далее рассматриваются особенности распараллеливания каждого из этапов.

```
G_0 = {G_0^0}
for k = 1,...,N :
    G_k = { }
    for G_{k-1}^i in G_{k-1} :
        G_k^i = { }
        for (y_0, y_{k-1}) in G_{k-1}^i :
            y_k = F_k(y_{k-1})
            G_k^i += (y_0, y_k)
        G_k += G_k^i
    for G_k^i in G_k :
        if eval_error(G_k^i) > ε and not is_leaf(G_k^i) :
            if all(eval_error(G_k^{i_s}) < ε for G_k^{i_s} in childs(G_k^i)) :
                G_k -= childs(G_k^i)
    for G_k^i in G_k :
        if eval_error(G_k^i) > ε and is_leaf(G_k^i) :
            ({G_{k-1}^{i_1}, G_k^{i_1}}, {G_{k-1}^{i_2}, G_k^{i_2}}) = build_split(G_{k-1}^i)
            G_{k-1} += {G_{k-1}^{i_1}, G_{k-1}^{i_2}}
            G_k += {G_k^{i_1}, G_k^{i_2}}
```

2.3.1. Структуры данных

Для эффективного использования технологии CUDA все данные желательно хранить в массивах. Отчасти это связано с тем, что для получения максимальной производительности каждые 32 потока, идущие подряд и образующие варп, должны выполнять всегда одну и ту же инструкцию и обращаться сразу к непрерывному участку в памяти [84]. Если это не выполняется, то возникает существенное уменьшение скорости работы. Поэтому динамические структуры данных, такие как деревья, на практике хранятся в массивах.

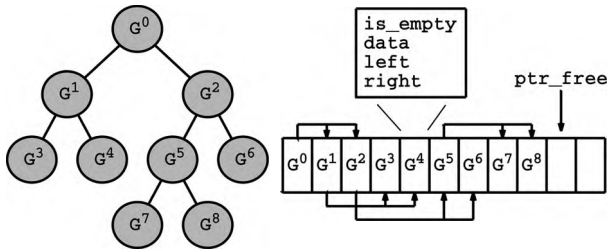

Рис. 32. Реализация дерева на массиве

Обычно при хранении дерева в массиве для каждой вершины четко определяются позиции дочерних вершин. Однако такой подход эффективен только тогда, когда дерево сбалансированное. В рассматриваемом случае в процессе работы алгоритма получающееся kd-дерево может быть сильно несбалансированным, что приводит к существенной фрагментации данных и избыточному потреблению памяти. Поэтому в структуре вершины дополнительно хранятся два индекса, обозначающие нахождение дочерних вершин. Добавление новых вершин всегда осуществляется в конец массива (рис. 32). При таком способе хранения фрагментация данных неизбежна, так как дерево часто перестраивается. В определенный момент, когда количество пустых ячеек становится сопоставимым с количеством занятых, запускается процесс дефрагментации, который выполняет уплотнение данных и перевычисляет все индексы. В его основе лежит один из фундаментальных алгоритмов GPU, которым является compact [85]. Суть алгоритма compact заключается в построении бинарного вектора признаков и нахождении префиксной суммы по

нему с помощью алгоритма scan. Альтернативным решением является хранение списка свободных вершин, но в общем случае оно не позволяет избежать фрагментации данных и не сохраняет последовательность добавления вершин.

Для реализации механизмов одновременного доступа ко всем узлам интерполяционных сеток и для устранения дублирования узлов в разных вершинах (рис. 33) они хранятся отдельно от kd-дерева. Для их хранения может подойти как хеш-таблица [86], размещенная в массиве, так и отсортированный массив. В обеих структурах для уменьшения запросов к глобальной памяти ключ хранится в отдельном от данных массиве, что позволяет при поиске не обращаться к связанным данным [87]. Если ограничиться только требованием эффективного доступа к элементам с GPU, то при $m = 1$ отсортированный массив работает быстрее хеш-таблицы в среднем в два раза. При $m = 2$ обе структуры данных уже примерно одинаковы, а при $m > 2$ хеш-таблица становится существенно более эффективной. Одна из причин такого поведения — это то, что узлы, находящиеся близко в пространстве, в отсортированном массиве оказываются в различных участках памяти.

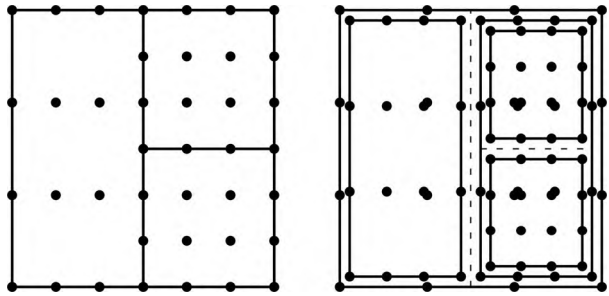

Рис. 33. Иллюстрация дублирования узлов интерполяционных сеток

Остановимся более подробно на реализации хеш-таблицы (рис. 34). Так же как и при реализации дерева, здесь всегда выполняется добавление элементов в конец массива с периодическим выполнением их уплотнения. Во-первых, такой подход позволяет сохранять последовательность вставки, а во-вторых, в процессе работы алгоритма вставка происходит пачками узлов, принадлежащих одной вершине.

При этом желательно, чтобы все узлы из одной пачки располагались в одном непрерывном участке памяти. В качестве хеш-функции от вектора вещественных значений используется следующая формула:

$$hash(x) = \left(\sum_{i=1}^{m} \left[\left[x_i / \varepsilon_0 + \frac{1}{2} \right] \bmod p \right] \times p^{m-i} \right) \bmod q,$$

где ε_0 — параметр дискретизации; p, q — простые числа.

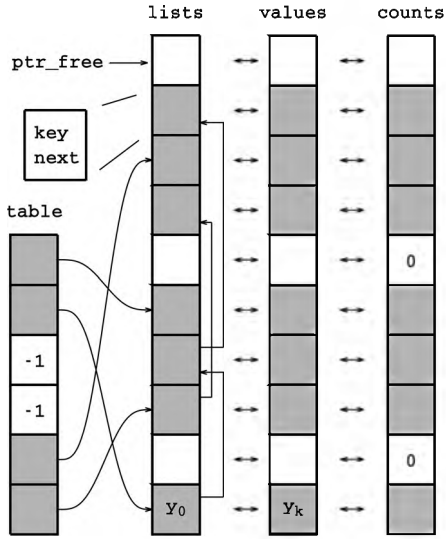

Рис. 34. Реализация хеш-таблицы на массивах

Для получения хорошего распределения значений хеш-функции дополнительно используется преобразование, которое было взято из OpenJDK6 — структуры HashMap:

```
h := hash(x);
h := h^ (h >> 20) ^ (h >> 12);
h := h^ (h >> 7) ^ (h >> 4).
```

Вместо kd-деревьев можно использовать и другие деревья разбиения пространства, например, дерево квадрантов в двухмерном случае или восьмеричное дерево — в трехмерном [88]. С одной стороны, они упрощают процесс уплотнения сетки, так как здесь возможно обой-

тись без перебора вариантов разбиения. Однако, с другой стороны, при каждом дроблении вместо двух новых вершин создаются сразу 2^m вершин, и зачастую это является неоправданным. Если предположить, что соответствующую область пространства действительно нужно разбить на 2^m частей, чтобы погрешность стала приемлемой, то в таком случае в kd-дереве создадутся $2\left(2^m-1\right)$ новых вершин. При использовании равномерных сеток интерполяционные узлы в дочерних вершинах будут полностью дублировать интерполяционные узлы в родительских. По этой причине даже двукратное превышение количества вершин несущественно отразится на производительности. В целом использование kd-дерева представляется оптимальным вариантом с точки зрения вычислительных затрат.

2.3.2. Обновление узлов интерполяционных сеток

Перед этапом распараллеливания необходимо проанализировать, какой может быть достигнут прирост производительности. Первый этап алгоритма представляет собой набор полностью независимых друг от друга задач, которые можно выполнять параллельно. Если F_k — достаточно тяжелая в вычислительном плане функция (например, представляет собой решатель ОДУ из нескольких тысяч уравнений), то на ее фоне операции по работе с деревом несущественны по трудозатратам, и вопрос об их распараллеливании отпадает. Зачастую в таком случае использование в лоб технологии CUDA, когда каждый поток вычисляет одну F_k, является неэффективным. Во-первых, процесс вычисления $F_k(y_k)$ в зависимости от y_k может сильно отличаться, что приведет к так называемой дивергенции потоков, когда часть потоков одного варпа идет по одной ветке вычислений, а часть — по другой. Во-вторых, может потребоваться большое количество переменных, и, как следствие, произойдет спиллинг регистров (часть переменных переместится из регистровой памяти в локальную) и скорость доступа упадет в сотни раз. В таком случае технология CUDA используется на уровне вычисления одной F_k.

В случае, когда F_k не требует значительных вычислительных ресурсов, например как с дискретной динамической системой (когда F_k — явно заданная функция), есть смысл проводить распаралле-

ливание операций над деревом и преимущественно использовать технологию CUDA.

В обоих случаях, при наличии вычислительного кластера, дополнительно можно применить технологию MPI для распределения задач по вычислительным машинам, а на самих машинах задействовать технологию CUDA.

2.3.3. Вычисление погрешности интерполяции

Вычисление погрешности интерполяции в конкретной вершине можно представить в виде последовательности действий:

1. Умножить значения, хранящиеся в узлах интерполяционных сеток, на соответствующие значения предвычисленных базисных полиномов Лагранжа (далее — интерполяционные коэффициенты).
2. Просуммировать значения, полученные в первом пункте, тем самым получив приближенные решения в тестовых точках.
3. Вычислить погрешность в каждой тестовой точке.
4. Найти максимальное значение погрешности из пункта 3.

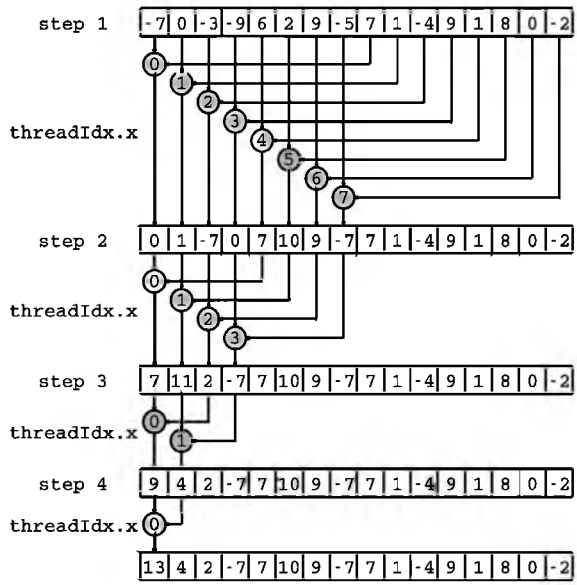

Рис. 35. Алгоритм reduction

Перемножение (п. 1) и вычисление погрешности (п. 3) распараллеливается без каких-либо сложностей, так как эти задачи полностью независимые и каждый поток обрабатывает свой элемент.

Для выполнения суммирования и поиска максимального значения применятся еще один из фундаментальных алгоритмов GPU, а именно reduction. Его суть отражена на рис. 35. Вместо операции сложения может использоваться любая другая ассоциативная бинарная операция, в частности «максимальный/минимальный из двух».

Если интерполяционных коэффициентов немного или каждый из них можно представить в виде произведения коэффициентов одномерной интерполяции, то для их хранения используется константная память, обладающая высокой скоростью доступа за счет механизма кеширования.

2.3.4. Разбиение и удаление вершин

Процесс разбиения вершин происходит в рамках перечисленных далее этапов, каждый из которых распараллеливается.

1. Определение всех вершин, подлежащих разбиению. Если таковые имеются, осуществляется переход ко второму пункту.
2. Построение всех возможных вариантов разбиений.
3. Применение ко всем узлам полученных вершин отображения F_k.
4. Вычисление погрешности интерполяции.
5. Выбор оптимальных вариантов. Переход к пункту 1.

С помощью алгоритма compact вычисляются положения вершин в массиве, для которых погрешность интерполяции больше заданного числа. Далее для каждой вершины строятся m вариантов ее разбиения. Для построения одного варианта необходимо определить значения, которые сопоставляются с узлами новых сеток, то есть по одной интерполяционной сетке следует построить две. Для вычисления одного узла в новой сетке все узлы старой сетки умножаются на интерполяционные коэффициенты и суммируются. Здесь точно так же, как и при вычислении погрешности, значения интерполяционных коэффициентов предвычисляются заранее и, если возможно, размещаются в константной памяти. После построения всех вариантов разбиения для всех вершин они переносятся на текущий временной

слой, где для них выполняется оценка погрешности. Из вариантов оставляются наиболее приемлемые, остальные удаляются.

Процесс удаления вершин является менее ресурсозатратным по сравнению с разбиением. Для каждой вершины проверяется, являются ли ее потомки листьями, в которых погрешность интерполяции меньше, чем некоторое число. Если это так, то потомки удаляются путем выставления флага is_empty, а соответствующая вершина становится листом. Данная операция выполняется несколько раз для того, чтобы при необходимости удаления целых поддеревьев не использовать рекурсию. Процесс останавливается, когда в дереве не остается вершин, подлежащих удалению. Сверху количество итераций ограничено высотой kd-дерева.

В целом при распараллеливании работы с интерполяционной сеткой следует учитывать количество содержащихся в ней узлов. Если оно невелико, то распараллеливание на уровне одной вершины может быть неэффективно. Также если все узлы помещаются в разделяемую память, то распараллеливание выполняется одним способом, если не помещаются, то другим. Конкретная реализация зависит от таких параметров решаемой задачи, как n, m, p и F_k.

2.4. ОПИСАНИЕ ПРОГРАММНОГО КОМПЛЕКСА

Программный комплекс [62] представляет собой набор библиотек на языке C++/CUDA, включающих в себя:

- Алгоритм адаптивной интерполяции на основе kd-дерева.
- Модуль подготовки данных для их визуализации.
- Интервальную арифметику. Интервальные векторы и матрицы.
- Интервальные методы линейной алгебры.
- Интервальные и классические методы оптимизации.
- Интервальные методы рядов Тейлора и их модификации.
- Модель Тейлора.

Программирование под графические процессоры обладает рядом специфических особенностей. В частности, при использовании ООП становится практически недоступной одна из ключевых его частей, а именно полиморфизм. Дело в том, что при копировании объекта в память видеокарты все указатели, которые содержатся в таблице

виртуальных методов, становятся некорректными, и при попытке вызвать соответствующие методы возникает ошибка. Поэтому здесь повсеместно используются шаблоны.

Далее приведем описание основных структур и методов, с помощью которых пользователь может взаимодействовать с kd-деревом.

Данный класс изначально проектировался как интерфейс. Он определяет сигнатуру метода, вычисляющего преобразование F_k:

```
class i_update {
public:
     __host__ __device__ void update(double *x,
                                const double *x0, int n) {
     }
};
```

Класс-интерфейс действия:

```
class i_action {
public:
     virtual __host__ void action(const double *x_key,
                               const double *x, int n) = 0;
};
```

С помощью конкретной реализации данного интерфейса можно «пройтись» по всем узлам интерполяционных сеток, содержащихся в kd-дереве, и применить метод action к каждому из них.

Зачастую при моделировании физических динамических систем те параметры, которые действительно представляют интерес, выражаются через некоторую функцию относительно фазовых переменных. Например, при моделировании процесса горения интерес представляют температура и давление, но в явном виде они в систему ОДУ не входят. Для того чтобы для них построить интервальную оценку, существует следующий класс-интерфейс:

```
template < int n_new >
class i_reform {
public:
   virtual   host   void reform(double x_new[n_new],
```

```
                        const double *x, int n) const = 0;
   virtual __host__ void reformd(double x_new[n_new],
                const double *x, int n, int id) const = 0;
   virtual __host__ void reformd2(double x_new[n_new],
     const double *x, int n, int id1, int id2) const = 0;
};
```

В методе reform определяется переход от фазовых переменных к интересующим пользователя параметрам. И уже при построении интервальной оценки методы оптимизации ищут экстремумы для этих параметров. В случае использования методов оптимизации первого или второго порядка пользователю необходимо определить еще два дополнительных метода — reformd и reformd2, задающих соответственно первые и вторые производные новых параметров по фазовым переменным.

Основной класс, представляющий реализацию kd-дерева:

```
template < int n, int dim, int order >
class t_kd_tree {
public:
     ...
     __host__ void update(T &fun, double eps);
     __host__ __device__ void eval(double x[n],
                                   const double p[dim]);
     __host__ void interval(double xd[n], double xu[n]);
     ...
};
```

где n — количество уравнений; dim — количество интервальных начальных условий; order — порядок интерполяционного полинома по каждому измерению. В этом классе содержится более 30 разных методов, позволяющих выполнять различные действия, но базовых всего три:

- `void update(T &fun, double eps)` — «перемещает» дерево на следующий временной слой с помощью объекта-преобразования fun (в нем должен быть реализован метод из класса i_update). Аргумент eps определяет точность.

- `void eval(double x[n], const double p[dim])` — для точечных значений интервальных параметров p возвращает решение x.
- `void interval(double xd[n], double xu[n])` — вычисляет интервальную оценку решения.

Следующий класс представляет собой обертку над kd-деревом для визуализации результатов:

```
template < int n, int dim, int order >
class t_kd_tree_writer {
public:
    ...
    void print_line(FILE *fp, const double p0[dim],
        const double p1[dim], int *projs, int n_projs);
    ...
}
```

Основная его задача заключается в том, чтобы подготовить текстовые файлы с данными для последующего их отображения в gnuplot'е. В классе реализовано множество разнообразных методов, в частности методы, которые отрисовывают разнообразные сетки, строят температурные карты, выполняют проецирование множеств на гиперплоскости и т.д. Все иллюстрации в работе выполнены с его помощью.

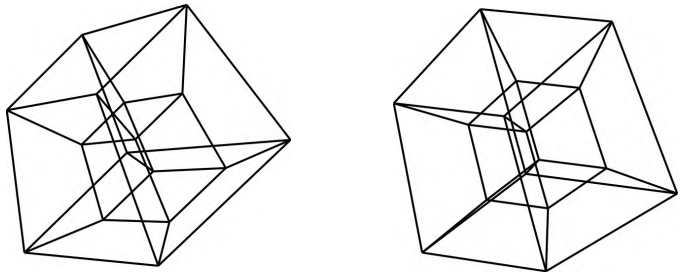

Рис. 36. Проекции вращающегося гиперкуба

Один из базовых методов класса — это print_line. Он соединяет прямой линией две точки, p1 и p2, в пространстве интервальных начальных условий. Естественно, в фазовом пространстве, в котором

находится множество решений, эта линия будет отнюдь не прямой. В общем случае она будет представлять собой кривую в многомерном пространстве. Для того чтобы ее отобразить в двухмерном или трехмерном пространстве (которое задается аргументами projs и n_projs), выполняется последовательность ортографических проекций [89]. На рис. 36 представлен гиперкуб, построенный таким образом.

2.5. РЕЗУЛЬТАТЫ РАСЧЕТОВ

Проведем решение некоторых задач на центральном процессоре и на графическом процессоре и сравним время работы отдельных этапов алгоритма. Характеристики CPU: Intel Core i7-3770 3.40GHz, оперативная память — 16GiB DDR3 1333MHz. Характеристики GPU: NVidia GeForce GTX 760, 2GiB, 1152 cuda cores.

Для интегрирования систем ОДУ использовался метод Рунге–Кутта четвертого порядка с шагом интегрирования 10^{-3}. Шаг дискретизации $\tau = t_{k+1} - t_k$ задавался постоянным и равным 5×10^{-2}. Он определяет промежуток, через который выполняется оценка погрешности и перестроение kd-дерева. Порядок интерполяционного полинома $p = 4$. Относительная точность — 10^{-5}.

Рассмотрим модель Лотки–Вольтерры с интервальными начальными условиями и одним интервальным коэффициентом. Задача Коши имеет вид [90]:

$$\begin{cases} x' = 4x - \dfrac{5}{4}xy - \alpha x^2, \\ y' = -2y + \dfrac{1}{2}xy - \dfrac{1}{20}y^2, \\ x(0) \in [4, 5],\ y(0) \in [2.8, 3.2], \\ t \in [0, 50], \end{cases} \tag{2.1}$$

где $\alpha \in [-0.05, 0.05]$.

Данная модель описывает взаимодействия типа «хищник — жертва». Особенностью системы является то обстоятельство, что при $\alpha < 0$ имеется неустойчивый фокус и амплитуда колебаний численности видов растет, а при $\alpha > 0$ — уже устойчивый фокус и со временем состояние системы стремится к стационарному.

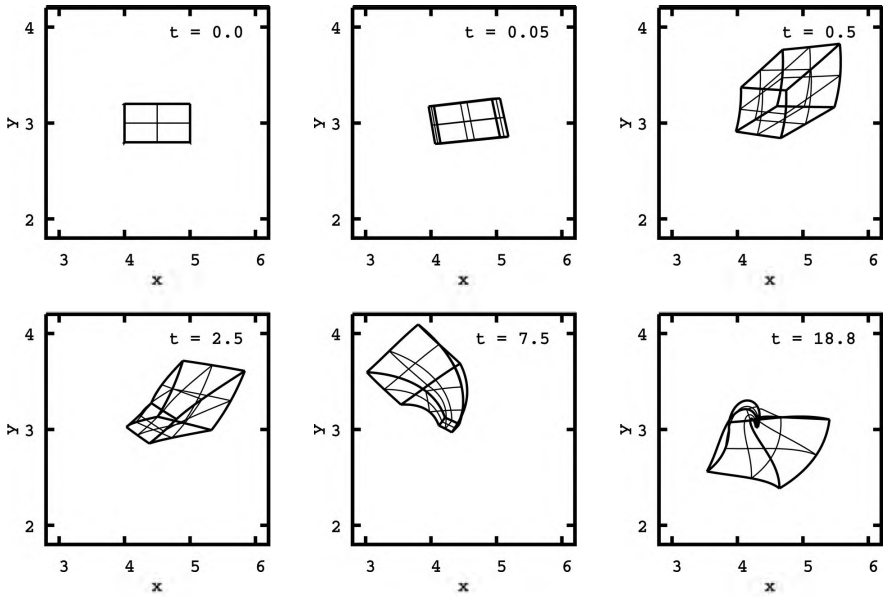

Рис. 37. Множество решений системы (2.1) в различные моменты времени

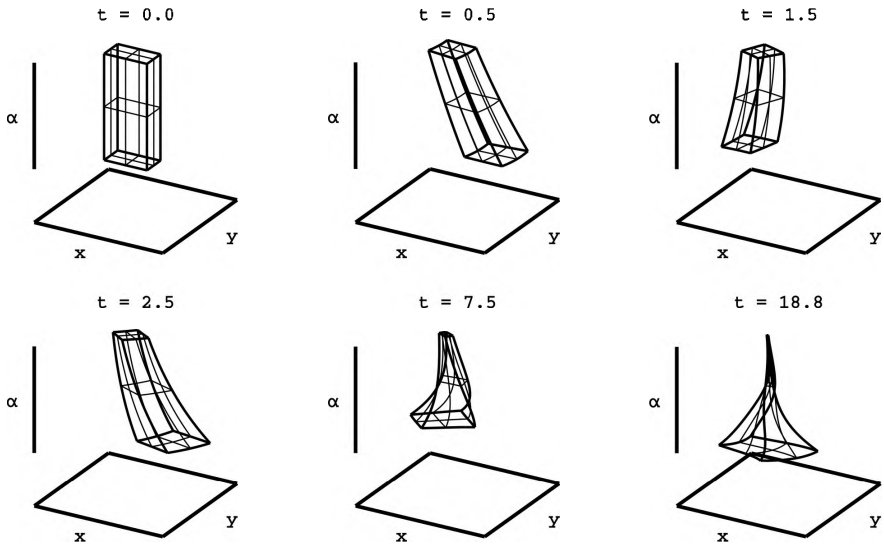

Рис. 38. Множество решений системы (2.1) в трехмерном пространстве

На рис. 37 и 38 представлено множество решений системы (2.1) в различные моменты времени. Здесь четко наблюдается следующая картина: часть множества устремляется в точку, что соответствует устойчивому фокусу, а часть множества увеличивается в размерах, что соответствует неустойчивому фокусу. Такое поведение системы приводит к постоянному увеличению плотности сетки по α (рис. 39). Из-за того что неопределенность присутствует в параметрах, множество решений на фазовой плоскости является лишь проекцией трехмерного множества (см. рис. 38) на двухмерное пространство. Дополнительному измерению соответствует интервальный параметр α.

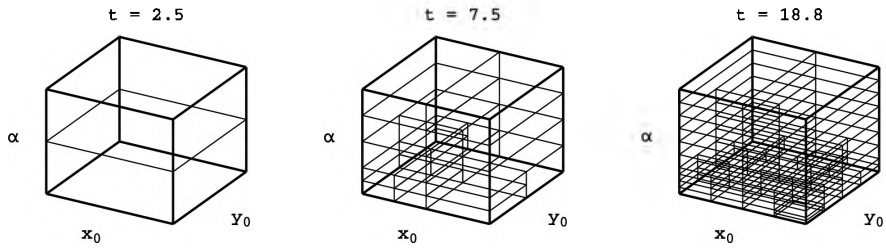

Рис. 39. Разбиения пространства, получающиеся в процессе решения системы (2.1)

В табл. 2.1 приводятся измерения времени работы каждого из этапов программной реализации алгоритма. Большая часть времени затрачивается на вычисление F_k. Так как этот этап алгоритма представляет собой набор решений полностью независимых друг от друга задач, то наибольший прирост скорости от распараллеливания наблюдается именно здесь: примерно в 30 раз. Вычисление погрешности и перестроение kd-дерева уже требуют некоторого взаимодействия между потоками и выполняются не так просто, как вычисление F_k, поэтому здесь ускорение от распараллеливания достигает только 8 раз.

Таблица 2.1

Сравнение времени работы этапов алгоритма при решении системы (2.1)

Этап / Процессор	Вычисление F_k, с	Вычисление погрешности, с	Перестроение kd-дерева, с
CPU	68.438	8.058	10.881
GPU	2.338	1.074	1.319

Рассмотрим следующую нелинейную систему ОДУ:

$$\begin{cases} x' = -y\left(\sqrt{\sin^2(x) + y^2} + 0.5\right)^{-1}, \\ y' = x\left(\sqrt{x^2 + \cos^2(y)} + 0.5\right)^{-1}, \\ x(0) \in [-5, 5], \\ y(0) \in [-5, 5]. \end{cases} \tag{2.2}$$

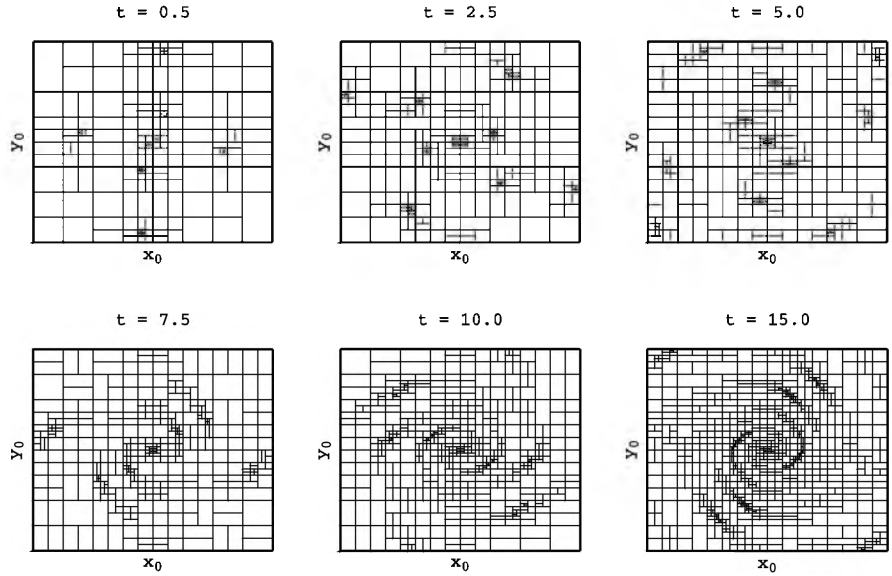

Рис. 40. Разбиения пространства, получающиеся в процессе решения системы (2.2)

Данная система ОДУ характеризуется тем, что в процессе ее интегрирования происходит частое перестроение kd-дерева, так как области сгущения и разрежения сетки перемещаются в пространстве (см. рис. 40). Этот факт отражается на затраченном времени работы соответствующего этапа алгоритма (табл. 2.2). Особенностью решения рассматриваемой задачи является также то обстоятельство, что в правой части ОДУ присутствуют «тяжелые» функции, такие как $\sqrt{}$, sin, cos, а это существенно сказывается на общем времени

работы. Прирост скорости практически в 100 раз при использовании распараллеливания возникает из-за того, что решение задачи позволяет полностью нагрузить графический процессор, в отличие от предыдущего примера, в котором получающиеся сетки были не столь плотными. На рис. 41 показано множество решений системы (2.2) в различные моменты времени.

Таблица 2.2

Сравнение времени работы этапов алгоритма при решении системы (2.2)

Этап / Процессор	Вычисление F_k, с	Вычисление погрешности, с	Перестроение kd-дерева, с
CPU	651.702	4.196	65.004
GPU	5.506	0.4688	0.977

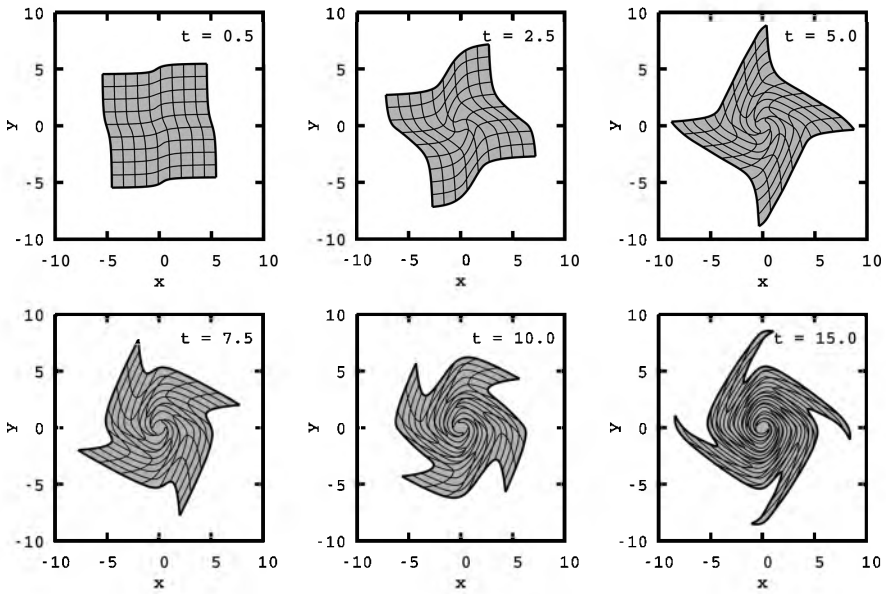

Рис. 41. Множество решений системы (2.2) в различные моменты времени

Отметим, что при оценке эффективности распараллеливания с использованием технологии CUDA классические метрики, такие как ускорение и эффективность, здесь не применяются, так как ядра GPU нельзя считать независимыми.

2.6. ЗАКЛЮЧЕНИЕ

В главе описываются различные аспекты распараллеливания и программной реализации алгоритма адаптивной интерполяции на основе kd-дерева для моделирования динамических систем с интервальными параметрами. Алгоритм состоит из нескольких этапов, каждый из которых распараллеливается с использованием технологии CUDA. Как показали вычислительные эксперименты, при интегрировании систем ОДУ большая часть времени тратится на первый этап работы алгоритма, который заключается в многократном вычислении одного и того же преобразования. В отличие от первого, распараллеливание второго и третьего этапа подразумевает некоторое взаимодействие между потоками, применение классических параллельных алгоритмов и использование различных видов памяти. В целом при проведении расчетов даже с использованием не самой современной видеокарты получено ускорение вычислений практически на два порядка по сравнению с вычислениями на центральном процессоре.

ГЛАВА 3. СРАВНИТЕЛЬНЫЙ АНАЛИЗ РАЗРАБОТАННЫХ МЕТОДОВ С ИЗВЕСТНЫМИ РЕАЛИЗАЦИЯМИ СУЩЕСТВУЮЩИХ

3.1. МЕТОДЫ РЯДОВ ТЕЙЛОРА

Рассмотрим интервальную задачу Коши:

$$\begin{cases} u' = f(u), \\ u(t_0) = u_0 \in \mathbf{u}_0, \end{cases}$$

где $f : \mathbb{R}^m \to \mathbb{R}^m$ — правая часть системы ОДУ, $\mathbf{u}_0 \in \mathbb{IR}^m$.

Интервальные методы рядов Тейлора [26, 41, 91] заключаются в разбиении исходного периода интегрирования на определенное число шагов; при этом каждый шаг интегрирования состоит из двух этапов. Первый этап заключается в определении текущего шага интегрирования и априорного интервала $\mathbf{v}_j$, гарантированно содержащего единственное решение $u(t)$ для всех $t \in [t_{j-1}, t_j]$ и $u_{j-1} \in \mathbf{u}_{j-1}$ (рис. 42).

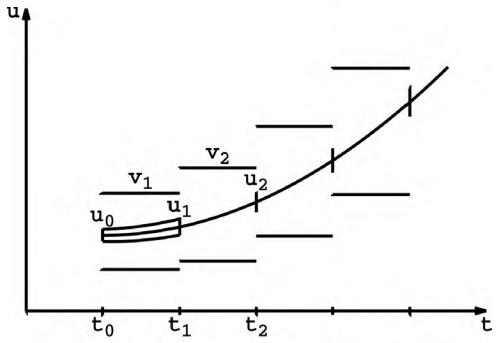

Рис. 42. Априорная оценка решения

На этом этапе часто применяют метод постоянного приближения первого порядка для автономных задач Коши, который заключается в следующем. Если h_j и $\mathbf{v}_j$ удовлетворяют условию $\mathbf{v}_j^1 = \mathbf{u}_{j-1} + [0, h_j] f(\mathbf{v}_j) \subseteq \mathbf{v}_j$, то задача Коши имеет единственное решение $u(t) \in \mathbf{v}_j^1$ для всех $t \in [t_{j-1}, t_j]$ и $u_{j-1} \in \mathbf{u}_{j-1}$.

На втором этапе с помощью априорного интервала $\mathbf{v}_j$ вычисляется более узкий интервал $\mathbf{u}_j$, гарантированно содержащий решение исходной задачи в точке t_j.

Каждый шаг интегрирования рассматривается как некоторое нелинейное преобразование из $\mathbb{R}^m$ в $\mathbb{R}^m$. При этом отделяется вычисление траектории перемещения области решения от производимых с ней деформаций под действием преобразования (для этого дополнительно вычисляется матрица Якоби преобразования).

Вычисление полных производных вплоть до k-го порядка имеет вид:

$$f^{[0]}(u) = u,\ f^{[k]}(u) = \left(\frac{\partial f^{[k-1]}}{\partial u} f \right)(u),\ k \geq 1,$$

вычисление матрицы Якоби преобразования:

$$\mathbf{S}_{j-1} = I + \sum_{k=1}^{n} \frac{h_j^k}{k!} J\left(f^{[k]}\right)\left(\mathbf{u}_{j-1}\right),$$

где I — единичная матрица; $J\left(f^{[k]}\right)$ — матрица Якоби для $f^{[k]}$. Для некоторой точки $\hat{u}_{j-1} \in \mathbf{u}_{j-1}$ получаем итоговую формулу:

$$u\left(t_j\right) \in \mathbf{u}_j = \hat{u}_{j-1} + \sum_{k=1}^{n-1} \frac{h_j^k}{k!} f^{[k]}\left(\hat{u}_{j-1}\right) + \mathbf{S}_{j-1}\left(\mathbf{u}_{j-1} - \hat{u}_{j-1}\right) + \mathbf{z}_j,$$

$$\mathbf{z}_j = \frac{h_j^{n+1}}{(n+1)!} f^{[n]}\left(\mathbf{v}_j\right).$$

Здесь эффект обертывания появляется в связи с членом $\mathbf{S}_{j-1}(\mathbf{u}_{j-1} - \hat{u}_{j-1})$.

Рассмотрим множество

$$\Sigma = \{S_{j-1}(u_{j-1} - \hat{u}_{j-1}) \,|\, S_{j-1} \in \mathbf{S}_{j-1}, u_{j-1} \in \mathbf{u}_{j-1}\}.$$

На каждом шаге интегрирования оно заменяется соответствующим прямоугольным параллелепипедом со сторонами, параллельными осям координат. За счет этого происходит паразитное увеличение множества решений.

Все методы семейства направлены на учет происходящих преобразований исходной области с дальнейшей их компенсацией на каждом шаге. Основная схема:

$$\hat{u}_0 = \mathrm{mid}(u_0),\ \mathbf{r}_0 = \mathbf{u}_0 - \hat{u}_0,\ B_0 = I,$$

$$\hat{u}_j = \hat{u}_{j-1} + \sum_{k=1}^{n-1} \frac{h_{j-1}^k}{k!} f^{[k]}(\hat{u}_{j-1}) + \mathrm{mid}\left(\mathbf{z}_j\right),$$

$$\mathbf{u}_j = \hat{u}_{j-1} + \sum_{k=1}^{n-1} \frac{h_{j-1}^k}{k!} f^{[k]}(\hat{u}_{j-1}) + \mathbf{z}_j + \left(\mathbf{S}_{j-1} B_{j-1}\right)\mathbf{r}_{j-1},$$

где $\mathbf{z}_j$ — локальная ошибка; $\mathbf{r}_j$ — глобальная ошибка:

$$\mathbf{r}_j = \left(B_j^{-1}\left(\mathbf{S}_{j-1} B_{j-1}\right)\right) r_{j-1} + B_j^{-1}\left(\mathbf{z}_j - \mathrm{mid}\left(\mathbf{z}_j\right)\right).$$

Все методы семейства различаются только способом вычисления матрицы B_j:

- Прямой метод Мура: $B_j = I$.
- Метод параллелепипедов (Еижгенраам, Лонер): $B_j = \mathrm{mid}\left(\mathbf{S}_{j-1} B_{j-1}\right)$.
- QR-метод (Лонер): $\mathrm{mid}\left(\mathbf{S}_{j-1} B_{j-1}\right) = QR$, $B_j = Q$.

Прямой метод Мура никак не компенсирует эффект обертывания. QR-метод Лонера отличается от метода параллелепипедов тем, что на каждом шаге область решения заключается в прямоугольный параллелепипед для более эффективного учета локальной ошибки $\mathbf{z}_j$.

Описанные выше методы значительно снижают эффект обертывания, особенно в однородных линейных системах ОДУ с постоянными коэффициентами. Однако при интегрировании нелинейных систем ОДУ эффект обертывания все равно проявляется, хотя и в меньшей мере, и начиная с определенного момента получаемые оценки решения становятся чрезмерно большими и неприемлемыми.

Матрица Якоби характеризует деформацию элементарных объемов пространства. В случае линейной системы ОДУ матрица Якоби постоянна и каждый элементарный объем области деформируется линейно и одинаково, как и в целом вся область. Поэтому область в процессе решения остается параллелепипедом. В случае нелинейной системы ОДУ матрица Якоби в каждой точке области разная. За счет использования интервальной арифметики получается уже интервальная матрица Якоби, содержащая в себе все возможные матрицы во всех точках области. При использовании такой матрицы для деформации заведомо закладывается то, что любой элементарный объем области

может быть деформирован любой вещественной матрицей Якоби. Это в конечном счете и ведет к паразитному эффекту.

В работе [26] были предложены несколько модификаций данных методов для снижения эффекта обертывания. Их идея заключается в том, чтобы вычислять матрицу Якоби не для всей области, а только для некоторой заведомо меньшей части или вообще только в одной точке. Использование такого подхода лишает свойства гарантированности, но, как показали численные эксперименты, большая часть решения все равно содержится в получаемой оценке.

3.1.1. Библиотека AWA

Библиотека AWA [28] разработана на Pascal-XSC в 1994 году Р. Лонером для решения ОДУ с гарантированной оценкой погрешности. Данная библиотека представляет собой реализацию QR-метода Лонера в качестве второго этапа и метода постоянного приближения для проверки существования и единственности решения в качестве первого этапа [41].

3.1.2. Библиотека VNODE-LP

Библиотека VNODE-LP [31] — это решатель интервальных систем ОДУ, разработанный в 2006 году Н.С. Недялковым на алгоритмическом языке C++. В отличие от традиционных решателей, которые вычисляют приблизительные решения, этот решатель доказывает, что существует единственное решение задачи, а затем строит строгие границы, которые гарантированно содержат его.

Библиотека является преемником библиотеки VNODE (Validated Numerical ODE), разработанной Н.С. Недялковым. Отличительной особенностью данного решателя является то, что он полностью посвящен «грамотному программированию». Грамотное программирование — это стиль написания программ, при котором программа рассматривается не как последовательность инструкций для компьютера, а как литературное произведение. Таким образом, программист при написании программы формально описывает действия, которые он ожидает от компьютера, вставляя в соответствующие места код на

языке программирования. Данная концепция предложена Д. Кнутом в 1984 году.

На первом этапе используется метод HOE, а во втором — метод Эрмита–Обрешкова. VNODE-LP имеет переменный шаг интегрирования и постоянный порядок.

3.2. МЕТОДЫ МОДЕЛИ ТЕЙЛОРА

Начиная с 1990-х годов профессором М. Берзом и его командой были разработаны методы модели Тейлора, которые сочетают интервальную арифметику с символьными вычислениями [39]. В этих методах основным типом данных является так называемая модель Тейлора, которая представляет собой полином некоторой степени с интервальным остатком:

$$p_n(x) + \mathbf{i}\,.$$

При использовании такого представления негативные эффекты, связанные с использованием интервальной арифметики, влияют только на интервальный остаточный член и полиномиальные члены порядка выше n, которые обычно очень малы.

Пусть $f : D \subset \mathbb{R}^m \to \mathbb{R}$ — $(n + 1)$ раз непрерывно дифференцируемая функция на открытом множестве D, содержащем интервальный вектор $\mathbf{x}$. Пусть x_0 — точка в $\mathbf{x}$, p_n — многочлен Тейлора n-го порядка функции f в окрестности x_0 и $\mathbf{i}$ — интервал такой, что:

$$f(x) \in p_n(x - x_0) + \mathbf{i},\ \forall x \in \mathbf{x}\,.$$

Пара $(p_n, \mathbf{i})$ называется моделью Тейлора n-го порядка для функции f в окрестности точки x_0 на $\mathbf{x}$.

Арифметические операции над моделями Тейлора определяются следующим образом:

$$T_1 + T_2 = (p_1 + p_2, \mathbf{i}_1 + \mathbf{i}_2),\ T_1 \cdot T_2 = (p_{1\cdot2}, \mathbf{i}_{1\cdot2}),$$

где $p_{1\cdot2}$ — часть полинома $p_1 \cdot p_2$, не превышая порядка n,

$$\mathbf{i}_{1\cdot2} = B(p_e) + B(p_1)\cdot\mathbf{i}_2 + B(p_2)\cdot\mathbf{i}_1 + \mathbf{i}_1\cdot\mathbf{i}_2\,,$$

где p_e — часть полинома $p_1 \cdot p_2$ порядка от $n + 1$ до $2n$, $B(p)$ — интервальная оценка полинома p на множестве $\mathbf{x}$.

Пример 3.1 [39]. Вычислим модель Тейлора второго порядка для экспоненты и косинуса на $\mathbf{x} = \left[-\frac{1}{2}, \frac{1}{2}\right]$. Разложим функции в ряд Тейлора в окрестности нуля:

$$e^x = 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 e^{\xi}, \ \cos(x) = 1 - \frac{1}{2}x^2 + \frac{1}{6}x^3 \sin(\xi),$$

где $x, \xi \in \mathbf{x}$. Получаем следующие модели Тейлора:

$$T_1(x) := 1 + x + \frac{1}{2}x^2 + [-0.035, 0.035], \ T_2(x) := 1 - \frac{1}{2}x^2 + [-0.010, 0.010].$$

Выполним их перемножение:

$$T_1(x)T_2(x) \subseteq \left(1 + x + \frac{1}{2}x^2\right)\left(1 - \frac{1}{2}x^2\right) + \left(1 + x + \frac{1}{2}x^2\right)[-0.010, 0.010] +$$

$$+\left(1 - \frac{1}{2}x^2\right)[-0.035, 0.035] + [-0.010, 0.010][-0.035, 0.035] \subseteq (1 + x) - \frac{1}{2}x^3 -$$

$$-\frac{1}{4}x^4 + [0.625, 1.625][-0.010, 0.010] + [0.875, 1][-0.035, 0.035] +$$

$$+[-0.004, 0.004] \subseteq 1 + x - [-0.063, 0.063] - [-0.016, 0.016] + [-0.202, 0.202] =$$

$$= 1 + x + [-0.281, 0.281].$$

В результате получена модель Тейлора для функции $e^x \cos x, x \in \mathbf{x}$:

$$e^x \cos x \in 1 + x + [-0.281, 0.281], \ x \in \mathbf{x}.$$

Решение интервальных систем ОДУ ищется в виде модели Тейлора относительно интервальных начальных условий. Все арифметические действия в методах заменяются соответствующими операциями над полиномами, в результате чего в конечный момент времени решение представляется в виде полинома относительно интервальных параметров и некоторого интервального остатка.

Пример 3.2 [39]. Рассмотрим систему ОДУ с интервальными начальными условиями:

$$\begin{cases} u' = v, \\ v' = u^2, \\ u(0) = 1 + u_0, v(0) = -1 + v_0, \end{cases}$$

где $u_0 \in [-0.05, 0.05]$ и $v_0 \in [-0.05, 0.05]$. Выполним два шага методом Эйлера с использованием моделей Тейлора 2-го порядка:

$$u(0.1) = 1 + u_0 + (-1 + v_0)0.1 = 0.9 + u_0 + 0.1v_0,$$

$$v(0.1) = -1 + v_0 + (1 + u_0)^2\, 0.1 = -0.9 + v_0 + 0.2u_0 + 0.1u_0^2,$$

$$u(0.2) = 0.9 + u_0 + 0.1v_0 + \left(-0.9 + v_0 + 0.2u_0 + 0.1u_0^2\right)0.1 =$$

$$= 0.81 + 1.02u_0 + 0.2v_0 + 0.01u_0^2,$$

$$v(0.2) = -0.9 + v_0 + 0.2u_0 + 0.1u_0^2 + (0.9 + u_0 + 0.1v_0)^2\, 0.1 =$$

$$= -0.819 + 1.018v_0 + 0.38u_0 + 0.2u_0^2 + 0.001v_0 + 0.02v_0u_0.$$

Уже начиная с третьего шага в процессе вычислений появятся степени больше второй, которые будут «уходить» в интервальный остаток. Полученное таким способом решение не является гарантированным, так как оно не учитывает погрешность метода Эйлера. В оригинальном методе происходит аналитическое разложение решения в ряд Тейлора и вычисление интервальной оценки остаточного члена. Поэтому полученное таким способом решение гарантированно содержит все решения исходной системы ОДУ.

В целом использование арифметики моделей Тейлора вместо классических интервальных арифметик существенно снижает эффект обертывания, особенно при интегрировании нелинейных систем ОДУ.

3.2.1. Библиотека COSY Infinity

Библиотека COSY Infinity [42] разрабатывается в Мичиганском университете профессором М. Берзом и его командой. Это система для проведения различных современных научных вычислений. Она состоит из следующих частей:

1. Набор расширенных и оптимизированных типов данных:
   - Дифференциальные алгебраические типы для многомерного исследования ОДУ, потоков и дифференциальных уравнений

в частных производных высокого порядка. Здесь имеется поддержка автоматического дифференцирования высокого порядка.

- Тип модели Тейлора для строгих вычислений высокого порядка. Разнообразные инструменты — в частности, инструменты для ограничения роста остаточного члена, инструменты для проверки ограничений и т.д.

2. Скриптовый язык COSYScript, в котором используются описанные выше типы. Он объектно ориентирован и поддерживает полиморфизм, имеет компактный и простой синтаксис, компилируется и выполняется на лету.

3. Интерфейсы для C, F77, C++ и F90 для использования типов во внешних программах.

4. Различные прикладные пакеты с использованием типов данных COSY, включая физику луча.

К сожалению, в текущей версии библиотеки COSY Infinity 10.1 (2018) модель Тейлора не поддерживается и при этом доступ к более ранним версиям отсутствует. В связи с этим выполнить прямое сравнение с данным программным комплексом не представляется возможным. Однако в работе [43] приводится сравнение библиотеки RiOT с COSY-VI на представительном наборе задач. Поэтому при сравнении с библиотекой RiOT будет выполнено косвенное сравнение с библиотекой COSY.

3.2.2. Библиотека RiOT

RiOT — это свободная программа на языке C++ для интегрирования систем ОДУ с интервальными начальными условиями, последняя версия — 2008 года [43]. В ней реализованы модель Тейлора, предложенная М. Берзом и К. Макино, а также механизм уменьшения накопления ошибок интегрирования Shrink Wrapping [49, 92], который позволяет выполнять интегрирование на длительные периоды. Идея механизма заключается в том, что за счет варьирования коэффициентов полинома удается как бы «спрятать» интервальный остаток в полином, где уже работает неинтервальная арифметика.

3.2.3. Библиотека FlowStar

Библиотека FlowStar [45] — это инструмент для достоверного моделирования гибридных динамических систем с интервальными параметрами, разработанный на языке C++, последняя версия — 2017 года. По значению интервальных параметров, натурального числа m и гибридной системы A, которая моделируется гибридным автоматом, FlowStar вычисляет конечный набор моделей Тейлора, содержащий в себе все состояния A, которые достижимы не более чем за m шагов.

Библиотека состоит в основном из двух модулей: вычислительной библиотеки и алгоритмов высокого уровня. В первый модуль входят реализации интервалов и интервальных арифметик, а также модель Тейлора. Во втором модуле реализованы методы, которые используются в анализе достижимости. Текущая версия FlowStar поддерживает гибридные системы, включающие в себя следующие компоненты:

- Непрерывные динамические системы, которые задаются с помощью ОДУ с интервальными начальными условиями и параметрами.
- Условия перехода, которые можно определять в виде системы полиномиальных неравенств.
- и т.д.

3.3. СРАВНЕНИЕ МЕТОДОВ И РЕАЛИЗАЦИЙ

Так как рассматриваемые библиотеки не распараллеливаются, их сравнение производится с реализацией алгоритма адаптивной интерполяции без использования технологии CUDA. Характеристика CPU и ОП: Intel(R) Core(TM) i5-4200M CPU @ 2.50GHz, 16 GiB 1600 MHz. Для всех библиотек параметры методов задавались одинаковыми: порядок 18, локальная абсолютная погрешность 10^{-11} (значение остаточного члена при разложении решения в ряд Тейлора) и начальный шаг $h_0 = 3 \times 10^{-2}$. Так как рассматриваемые далее несколько задач взяты из работы [43], такой выбор параметров соответствует приведенным в этой работе расчетам. Для алгоритма адаптивной интерполяции порядок задавался $p = 4$ и относительная погрешность 10^{-5}. Интег-

рирование неинтервальных ОДУ выполнялось методом Рунге–Кутты четвертого порядка с постоянным шагом $h = 10^{-3}$.

Модель Лотки–Вольтерры:

$$\begin{cases} x' = 2x - 2xy, \\ y' = -y + xy, \\ x(0) \in [0.95, 1.05], \\ y(0) \in [2.95, 3.05], \\ t \in [0, 100]. \end{cases} \tag{3.1}$$

На рис. 43 показано множество решений системы (3.1) в различные моменты времени. Для начальной точки $x = 1$, $y = 3$ период вращения равен $T = 5.488138468035$, о чем свидетельствует совпадение центров множеств в начальный момент времени и при $t = T$. Из-за того, что для остальных начальных точек, содержащихся в интервальных начальных условиях, период вращения другой, происходит вытягивание множества.

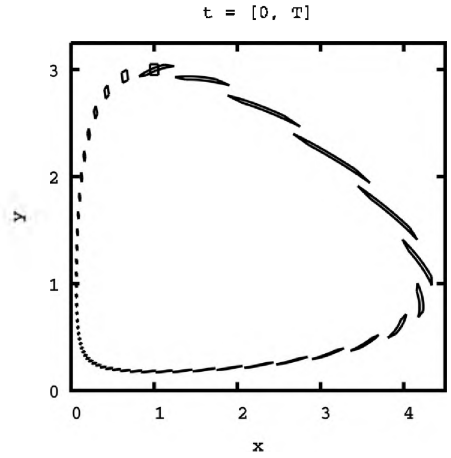

Рис. 43. Один период вращения множества решений системы (3.1)

В табл. 3.1 и на рис. 44 приведено сравнение результатов, полученных разными программными комплексами. Библиотеки AWA и VNODE-LP практически не компенсируют эффект обертывания,

и уже с момента $t > 4.5$ его влияние оказывается существенным и решения расходятся. Отметим, что по вычислительным затратам библиотека VNODE-LP наиболее эффективная.

Таблица 3.1

Сравнение результатов решения системы (3.1) в момент времени 0.8T

Библиотека	Время, с	x	y
Точное решение	—	[2.469047, 2.847741]	[0.244560, 0.315898]
Алгоритм адаптивной интерполяции	0.038	[**2.469047**, **2.847741**]	[**0.244560**, **0.315898**]
AWA	0.114	[**2**.207070, 3.111953]	[0.172401, **0.3**81625]
VNODE-LP	0.005	[**2**.243064, 3.075959]	[0.187871, **0.3**66156]
RiOT	23.82	[**2.46**8492, **2.84**8475]	[**0.24**2515, **0.31**6225]
FlowStar	68.87	[**2**.373701, 3.179943]	[**0.2**20549, **0.3**70295]

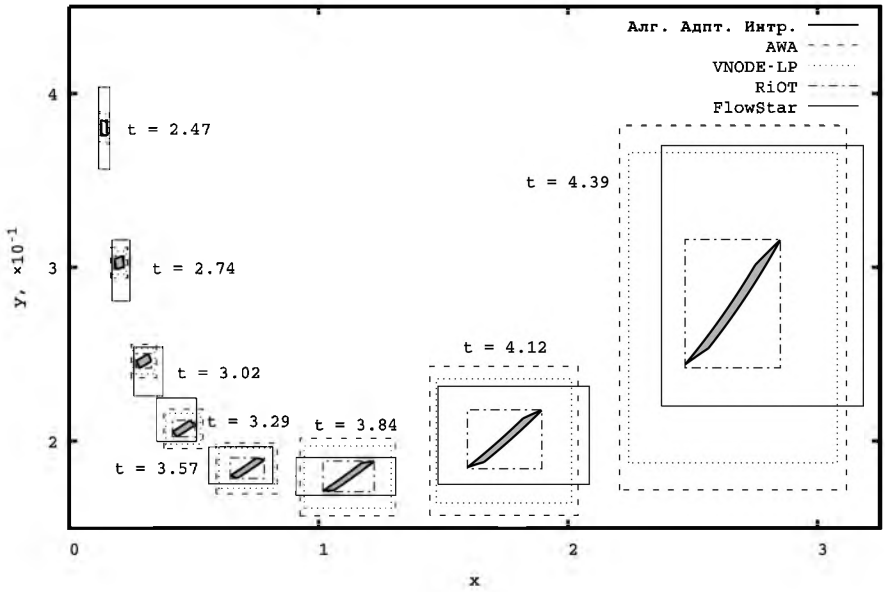

Рис. 44. Сравнение решений системы (3.1), полученных различными библиотеками, в различные моменты времени

В табл. 3.2 приведено сравнение с библиотекой COSY-VI. Здесь и далее время работы COSY-VI оценивается относительно библиотеки RiOT, так как в работе [43] имеется их сопоставление.

Таблица 3.2

Сравнение результатов решения системы (3.1) в момент времени T

Библиотека	Время, с	x	y
Точное решение	—	[0.816719, 1.240265]	[2.936455, 3.045758]
Алгоритм адаптивной интерполяции	0.039	[**0.816719**, **1.240265**]	[**2.936455**, **3.045758**]
RiOT	287.5	[**0**.798904, **1.240**280]	[**2.93**3777, **3.0**54377]
FlowStar	158.8	[0.557513, **1**.470005]	[**2**.869371, **3**.113960]
COSY-VI	≈ 2.96	[**0.816719**, **1.240265**]	[**2.93**5927, **3.045759**]

На рис. 45 и в табл. 3.3 приведено сравнение решений системы (3.1) в момент времени $t = 14.56$. Здесь наблюдается сильное завышение получаемых оценок решений.

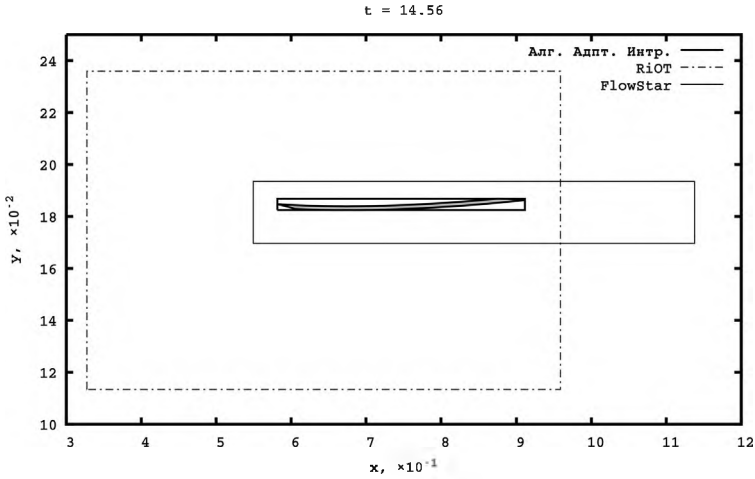

Рис. 45. Оценки решений системы (3.1) различными библиотеками

Таблица 3.3

Сравнение результатов решения системы (3.1) в момент времени 14.56

Библиотека	Время, с	x	y
Точное решение	—	[0.581638, 0.911206]	[0.182434, 0.186810]
Алгоритм адаптивной интерполяции	0.068	[**0.581638**, **0.911206**]	[**0.182434**, **0.186810**]
RiOT	903.7	[0.327651, **0.9**58718]	[0.113389, 0.235948]
FlowStar	460.4	[**0.5**49449, 1.137605]	[**0.1**69651, **0.1**93453]

На рис. 46 приведена иллюстрация решения, полученного библиотекой FlowStar. Интервальный остаток (i) существенно больше полиномиальной части решения, и уже при $t \approx 16$ библиотека аварийно завершает расчет.

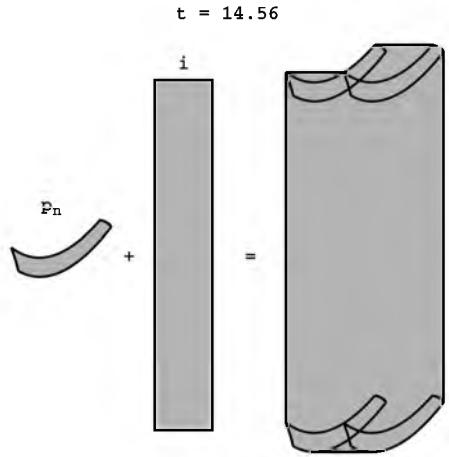

Рис. 46. Модель Тейлора. Решение системы (3.1), полученное FlowStar

Осциллятор Ван дер Поля:

$$\begin{cases} x' = y, \\ y' = \eta\left(1 - x^2\right)y - x,\ \eta > 0, \\ x(0) \in [2.999,\ 3.001], \\ y(0) \in [-3.001,\ -2.999], \\ t \in [0,\ 10], \end{cases} \tag{3.2}$$

где $\eta = 1$.

Особенностью данной системы является то, что в ней существует предельный цикл, к которому стремятся решения. В связи с этим изначально двухмерное множество решений вырождается в линию (рис. 47).

Библиотека VNODE-LP завершает расчет аварийно при $t \approx 20.5$ (рис. 48). Библиотека AWA проявляет себя немного хуже, и решения, получаемые ей, расходятся уже с момента времени $t \approx 17.4$ (рис. 49).

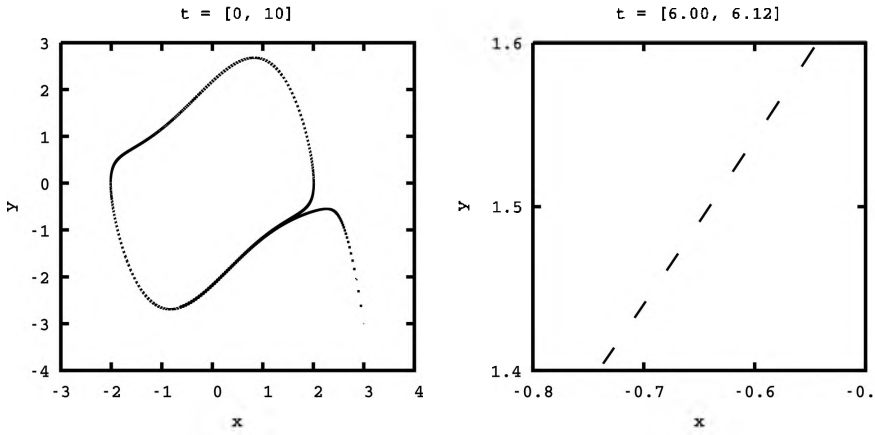

Рис. 47. Множество решений системы (3.2) в различные моменты времени

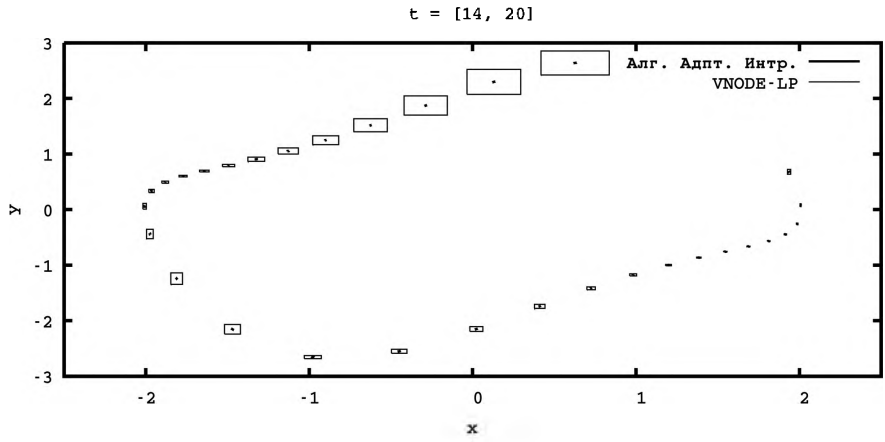

Рис. 48. Сравнение с библиотекой VNODE-LP

Для библиотеки RiOT в этой задаче использовались следующие параметры: порядок — 10, локальная абсолютная погрешность — 10^{-7}. На рис. 50 наблюдается скачкообразное увеличение интервальной оценки решений. Это связано с применением механизма Shrink Wrapping. Начиная с момента времени $t \approx 14$ получаемые интервальные оценки решений начинают расходиться, и уже при $t \approx 14.6$ библиотека аварийно завершает расчет.

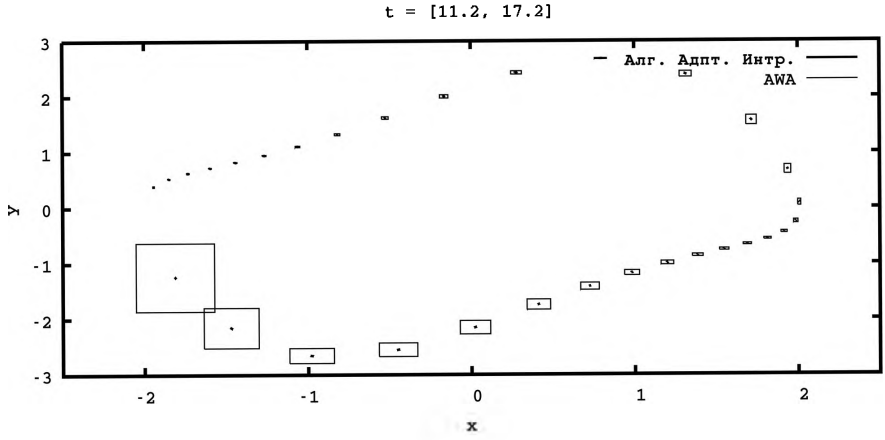

Рис. 49. Сравнение с библиотекой AWA

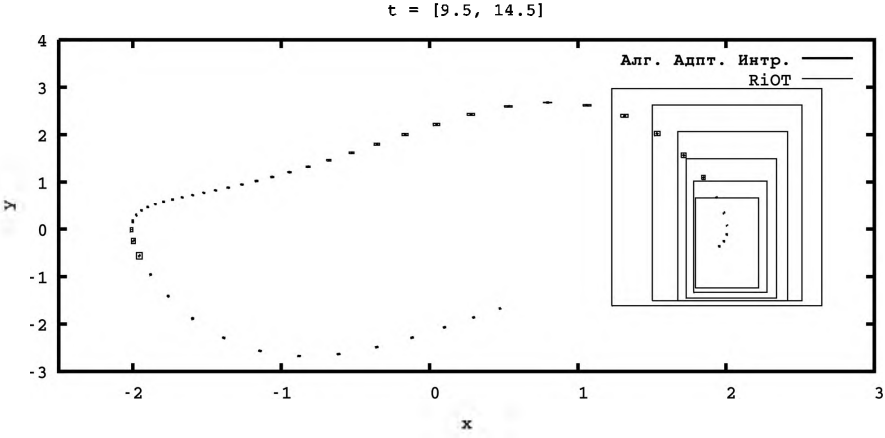

Рис. 50. Сравнение с библиотекой RiOT

На рис. 51 показаны интервальные оценки множества решений системы (3.2), полученные библиотекой FlowStar. Из всех рассмотренных библиотек данная библиотека смогла проинтегрировать систему ОДУ дальше всех по времени. Начиная с $t \approx 23$ эффект накопления локальных ошибок становится существенным, и решения расходятся (табл. 3.4 и 3.5). Отметим, что алгоритм адаптивной интерполяции

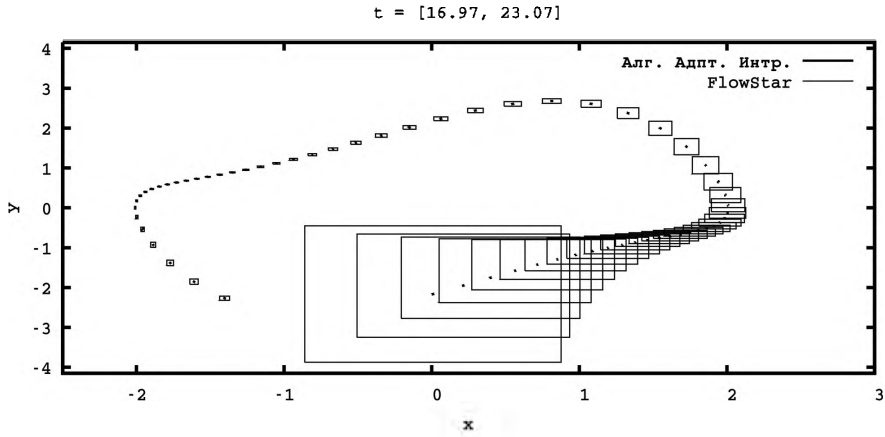

Рис. 51. Сравнение с библиотекой FlowStar

способен интегрировать данную систему ОДУ до любого момента времени.

Таблица 3.4

Сравнение результатов решения системы (3.2) в момент времени 10.0

Библиотека	Время, с	x	y
Точное решение	—	[–0.622979, – 0.604482]	[–2.637859, – 2.630639]
Алгоритм адаптивной интерполяции	0.023	[**–0.622979**, – **0.604482**]	[**–2.637859**, – **2.630639**]
AWA	0.256	[**–0.6**32841, – **0.5**94607]	[**–2.6**43362, – **2.6**25265]
VNODE-LP	0.008	[**–0.6**30462, – **0.5**96986]	[**–2.6**41948, – **2.6**26680]
RiOT	3.448	[**–0.62**3069, – **0.604**389]	[**–2.637**901, – **2.6306**08]
FlowStar	127.9	[**–0.6**32770, – 0.367207]	[**–2.6**79636, – **2.**460644]
COSY-VI	≈ 0.509	[**–0.6**82849 , – 0.544630]	[**–2.6**57389, – **2.6**03807]

Таблица 3.5

Сравнение результатов решения системы (3.2) в момент времени 22.0

Библиотека	Время, с	x	y
Точное решение	—	[1.435349, 1.441166]	[–0.830405, – 0.826522]
Алгоритм адаптивной интерполяции	0.041	[**1.435349**, **1.441166**]	[**–0.830405**, – **0.826522**]
FlowStar	300.9	[**1.**376780, **1.4**77120]	[**–0.8**70178, – **0.8**02331]

Рассмотрим систему ОДУ:

$$\begin{cases} x' = \eta\left(\dfrac{\sqrt{3}}{2}x + \dfrac{1}{2}y\right), \\ y' = \eta\left(-\dfrac{1}{2}x + \dfrac{\sqrt{3}}{2}y\right), \\ x(0) \in [-1, 1],\ y(0) \in [-1, 1], \\ t \in [0, 4.5], \end{cases} \tag{3.3}$$

где $\eta \in [-1,1]$.

На рис. 52 и 53 показано множество решений исходной системы ОДУ в различные моменты времени. Здесь наблюдается закручивание множества в спиралевидную воронку. Для алгоритма адаптивной интерполяции это приводит к постоянному увеличению плотности сетки по измерению η. В плоскости x,y при каждом конкретном значении η соответствующие сечения множества претерпевают линейные деформации растяжения и вращения.

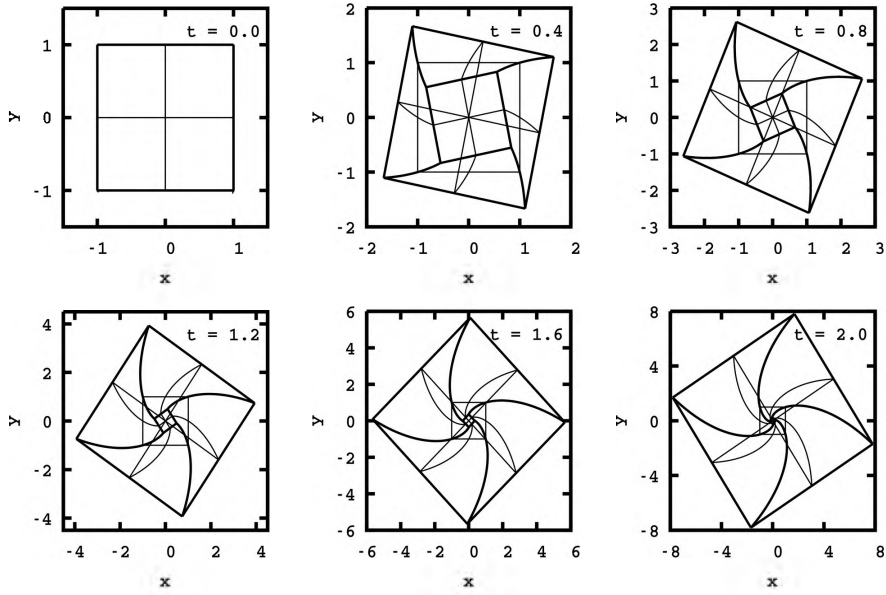

Рис. 52. Множество решений системы (3.3) в различные моменты времени

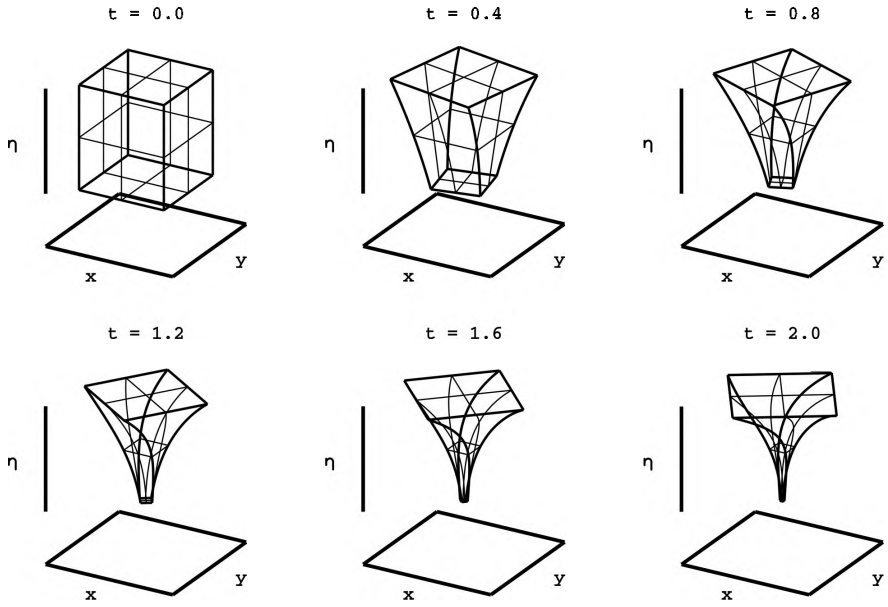

Рис. 53. Множество решений системы (3.3) в трехмерном пространстве

Библиотеки AWA и VNODE-LP работают быстрее всех (табл. 3.6), но при этом они дают очень завышенные оценки решений и спустя некоторое время аварийно завершают расчет.

Таблица 3.6

Сравнение результатов решения системы (3.3) в момент времени 0.8

Библиотека	Время, с	x	y
Точное решение	—	[–2.620102, 2.620102]	[–2.620102, 2.620102]
Алгоритм адаптивной интерполяции	0.055	[**–2.620102**, **2.620102**]	[**–2.620102, 2.620102**]
AWA	0.008	[–7.379424, 7.379424]	[–7.379424, 7.379424]
VNODE-LP	0.004	[–5.270461, 5.270461]	[–5.270461, 5.270461]
FlowStar	0.072	[**–2.6**42752, **2.6**42752]	[**–2.6**42752, **2.6**42752]

На рис. 54 представлено сравнение решений, полученных указанными библиотеками. В целом, как и в предыдущих примерах, библиотека FlowStar проявляет себя лучше, чем AWA и VNODE-LP, с точки зрения получаемых результатов. Но тем не менее из-за того,

что метод, реализованный в ней, подвержен накоплению ошибок, спустя некоторое время от начала интегрирования получаемые интервальные оценки начинают расходиться.

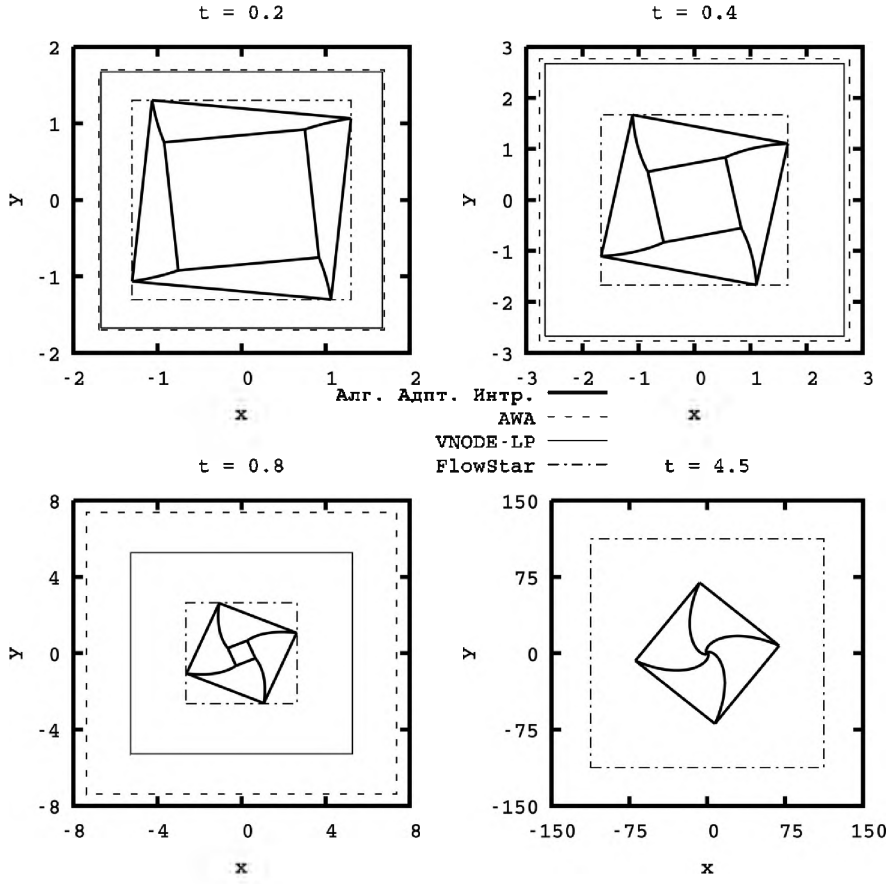

Рис. 54. Сравнение решений системы (3.3), полученных различными библиотеками, в различные моменты времени

3.4. ЗАКЛЮЧЕНИЕ

Выполнен обзор существующих библиотек и реализованных в них методов моделирования динамических систем с интервальными параметрами. Произведено сравнение полученных авторами результатов,

с результатами, полученными доступными программными библиотеками AWA, VNODE, COSY Infinity, RiOT и FlowStar. Сравнение на представительном наборе тестовых задач показывает превосходство алгоритма адаптивной интерполяции и его реализации с точки зрения точности и вычислительных затрат. За счет использования принципиально другого подхода к решению задач с интервальными параметрами алгоритм адаптивной интерполяции не подвержен накоплению ошибок, получает границы решений с контролируемой точностью и работает на порядки быстрее аналогов.

ГЛАВА 4. ПРИМЕНЕНИЕ АЛГОРИТМА АДАПТИВНОЙ ИНТЕРПОЛЯЦИИ ДЛЯ РЕШЕНИЯ ПРИКЛАДНЫХ И ИССЛЕДОВАТЕЛЬСКИХ ЗАДАЧ

4.1. БИФУРКАЦИИ И ХАОС

Согласно работе [93], термин «бифуркация» означает «раздвоение» и употребляется в качестве названия любого скачкообразного изменения, происходящего при плавном изменении параметров динамической системы. На практике реальные физические системы описываются дифференциальными уравнениями, содержащими в себе параметры, точные значения которых, как правило, неизвестны. Если уравнения оказываются структурно неустойчивыми, то есть поведение решений может качественно измениться при сколь угодно малом изменении правой части, то моделирование таких систем представляет собой нетривиальную задачу. Возникновение хаоса связано с особым свойством нелинейных систем экспоненциально быстро разводить решения в ограниченной области фазового пространства. Применение классических интервальных методов в этом случае становится практически невозможным, а символьно-интервальные методы требуют расщепления исходной области неопределенности на меньшие подобласти. Это связано с тем, что такие режимы сопровождаются очень резкими изменениями решений, аппроксимация которых представляет собой проблему.

Возможность определять наличие в динамической системе бифуркаций и хаоса является важным аспектом при исследовании динамических систем. В этой области существует достаточно много работ [65, 94, 95], посвященных как аналитическим подходам, так и численным. Существующие методы не всегда успешно справляются с данной задачей: некоторые ограничены размерностью пространства (не более двух), для некоторых важным является выбор начального приближения, для других требуется аналитическое представление динамической системы.

В окрестности точки бифуркации возникает особенность, которая не позволяет разложить решение в ряд по степеням параметров. Для

хаоса отличительной чертой является существенная зависимость от начальных условий. Это означает, что изначально близкие траектории спустя некоторое время кардинально разойдутся. В обоих этих случаях попытка построить функцию, интерполирующую зависимость решения от параметров или начальных условий, будет заканчиваться неудачей. На этой идее основывается описываемый здесь подход.

Для возможности идентификации особенностей в алгоритм адаптивной интерполяции вводится дополнительный параметр — минимальный размер ячейки разбиения. Если в процессе работы алгоритма возникают ячейки такого размера, это является признаком наличия в динамической системе неустойчивых состояний или особенностей.

Рассмотрим одномерную динамическую систему, описываемую уравнением

$$x' = \lambda x + x^3 - x^5,$$

где λ — внешний управляющий параметр. Для данной системы аналитически можно найти все точки бифуркаций и построить бифуркационную диаграмму (рис. 55). Здесь присутствует жесткая потеря устойчивости. Пунктирными линиями показаны неустойчивые состояния.

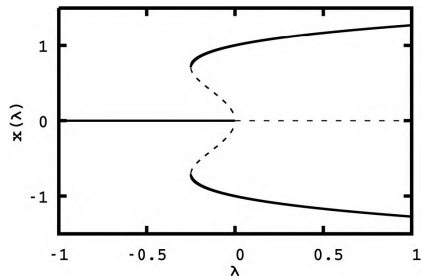

Рис. 55. Бифуркационная диаграмма

С помощью алгоритма адаптивной интерполяции решим следующую задачу Коши:

$$\begin{cases} x' = \lambda x + x^3 - x^5; \\ x(0) = x_0 \in [-2, 2], \end{cases}$$

где $\lambda \in [-1, 1]$.

На рис. 56 показана зависимость решения от интервальных параметров задачи.

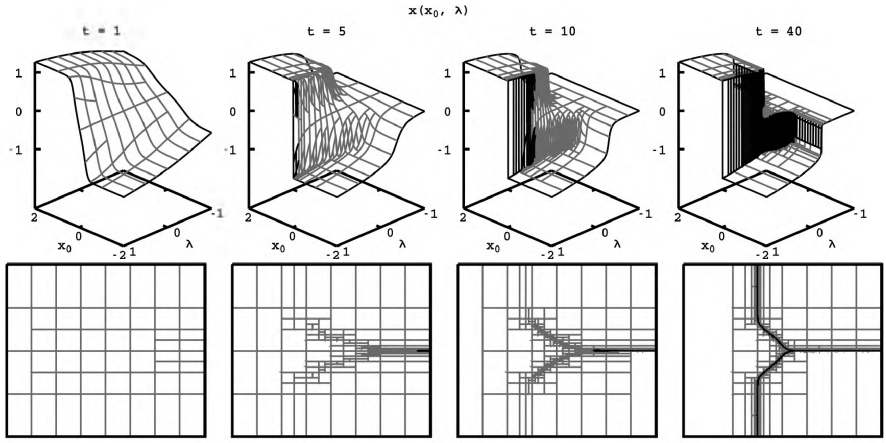

Рис. 56. Зависимость решения от начальных условий и параметров

На верхних рисунках показана зависимость решения от вещественных значений $x(0)$ и λ, на нижних рисунках изображены получающиеся разбиения области неопределенности в процессе работы алгоритма. Черным цветом выделены ячейки минимального размера.

4.1.1. Осциллятор Ван дер Поля

Одной из основных моделей для анализа периодических автоколебаний служит уравнение Ван дер Поля:

$$x'' - \left(\lambda - x^2\right)x' + x = 0 .$$

В работе [96] показано, что возбуждение автоколебаний происходит при $\lambda > 0$ и по мере увеличения λ происходит постепенный переход от слабонелинейных квазигармонических колебаний к релаксационным.

При $\lambda < 0$ на фазовой плоскости имеется единственное состояние равновесия типа «устойчивый фокус». При $\lambda > 0$ оно становится неустойчивым и в его окрестности возникает устойчивый предельный цикл, причем характерный размер предельного цикла увеличивается

пропорционально $\sqrt{\lambda}$. Такая бифуркация называется бифуркацией Андронова–Хопфа. Выполним численное исследование поведения системы в зависимости от параметра λ. Рассмотрим систему ОДУ:

$$\begin{cases} x' = y, \\ y' = \left(\lambda - x^2\right) y + x, \\ x(0) = 3,\ y(0) = 3, \\ t \in [0, 100], \end{cases} \tag{4.1}$$

где $\lambda \in [-1, 3]$. На рис. 57 ($\lambda \in [-1, 1]$) и 58 ($\lambda \in [1, 3]$) показано множество решений системы (4.1) в различные моменты времени.

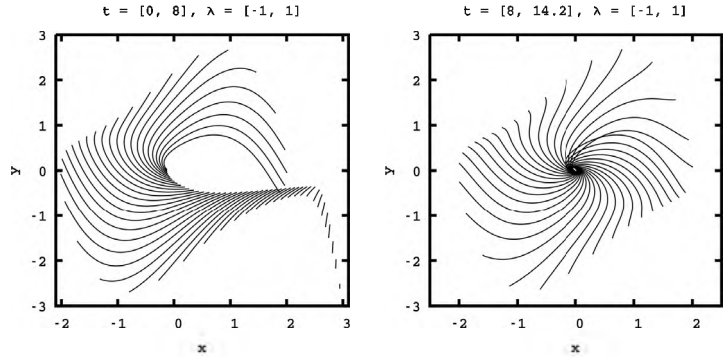

Рис. 57. Множество решений системы (4.1) в различные моменты времени ($\lambda \in [-1,1]$)

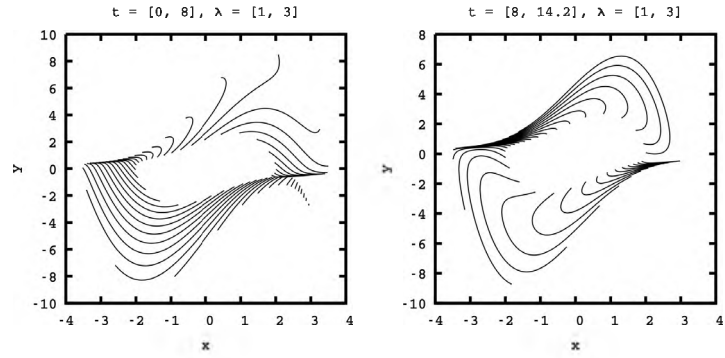

Рис. 58. Множество решений системы (4.1) в различные моменты времени ($\lambda \in [1,3]$)

В процессе интегрирования решение, будучи изначально точкой на фазовой плоскости, превращается в замысловатую кривую. Каждая такая кривая на графиках соответствует определенному моменту времени. Концы кривых, которые располагаются ближе к центру, соответствуют наименьшему значению параметра λ, а те, которые дальше, — наибольшему.

На рис. 59, 60 и 61 отражены зависимости решений от λ. Как и на предыдущих графиках, каждая кривая — это решение, полученное в определенный момент времени.

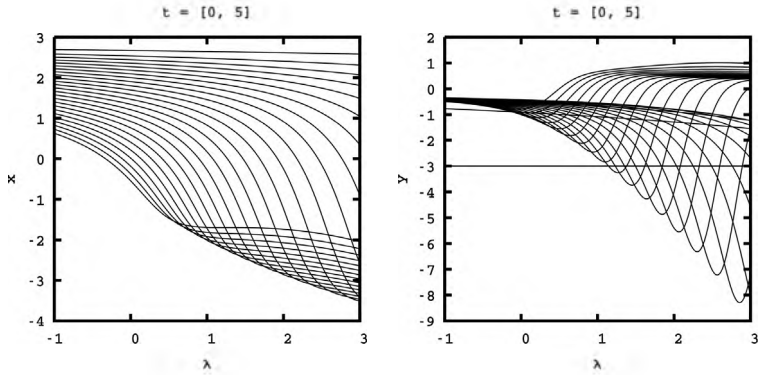

Рис. 59. Зависимость решений системы (4.1) от интервального параметра при $t \in [0, 5]$

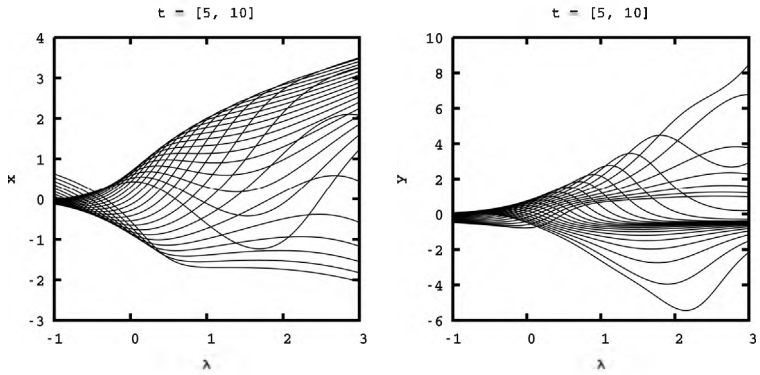

Рис. 60. Зависимость решений системы (4.1) от интервального параметра при $t \in [5, 10]$

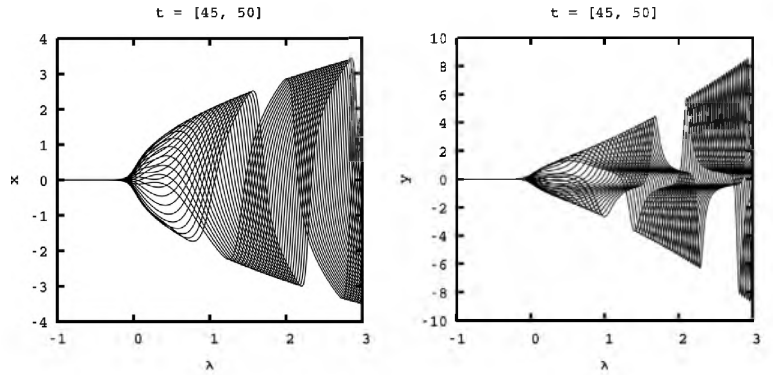

Рис. 61. Зависимость решений системы (4.1) от интервального параметра при $t \in [45, 50]$

В начальные моменты времени (см. рис. 59, 60) решения при $\lambda < 0$ еще не пришли в устойчивый фокус. Со временем (рис. 61) начинает выделяться точка бифуркации $\lambda = 0$. Слева от нее решения остаются на месте, в то время как справа наблюдаются периодические колебания.

Основной прием для обнаружения бифуркаций или хаоса заключается в анализе динамики перестроения kd-дерева. На рис. 62 отражена зависимость высоты kd-дерева от λ для двух моментов времени.

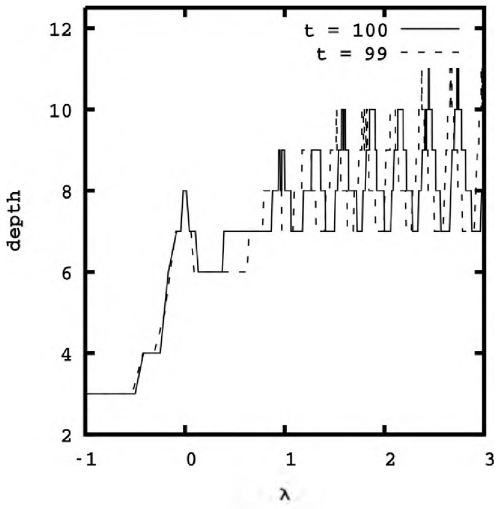

Рис. 62. Зависимость высоты kd-дерева от интервального параметра для двух моментов времени

При $\lambda = 0$ наблюдается четко выраженное стационарное уплотнение ячеек, так как решение в этой точке теряет свою «аналитичность», в пределе производная по параметру λ не существует. При этом характер уплотнения в этой задаче носит не критический характер: размер ячеек не достигает минимального значения, в отличие от предыдущего примера.

4.1.2. Осциллятор Дуффинга

Осциллятор Дуффинга — простейшая модель осциллятора с реактивной нелинейностью, которая задается уравнением

$$x'' + \alpha x' + \omega_0^2 \left(1 + x^2\right) = F(t),$$

где α — коэффициент диссипации; ω_0 — собственная частота; $F(t)$ — внешнее воздействие.

Согласно работе [97], данное уравнение можно получить, например, рассматривая колебания математического маятника при небольших углах отклонения (рис. 63,*а*), колебания грузика на пружине с нелинейной возвращающей силой, расположенного на горизонтальной поверхности (рис. 63,*б*), или при описании движения частицы в двух потенциальных ямах (рис. 63,*в*).

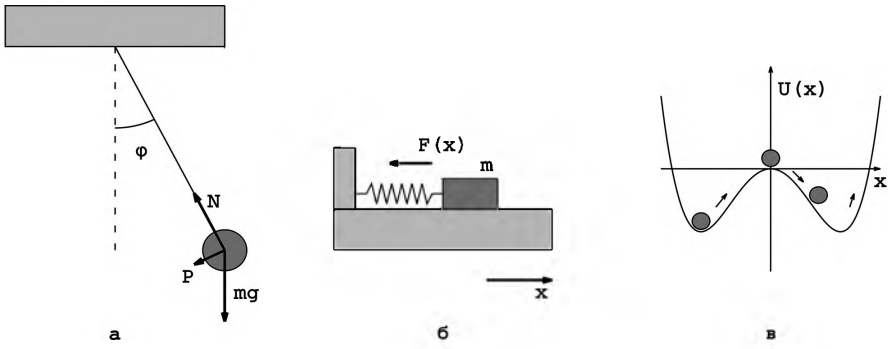

Рис. 63. Примеры колебательных систем

В работе [98] показано, что осциллятор Дуффинга представляет собой систему с двумя устойчивыми равновесными состояниями, которые разделяет единая сепаратриса, и возбуждением. При опре-

деленных параметрах вблизи сепаратрисы возникают хаотические решения. Такие системы являются предельными тестами для подобных алгоритмов, так как в процессе вычисления требуется постоянное уплотнение сетки.

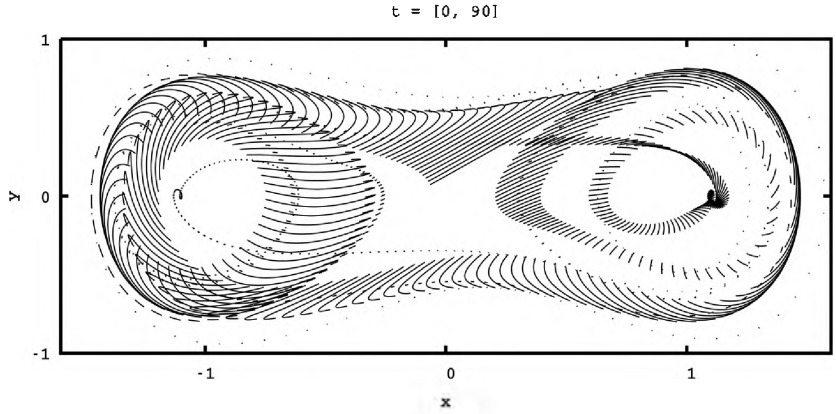

Рис. 64. Множество решений системы (4.2) в различные моменты времени ($t \in [0, 90]$)

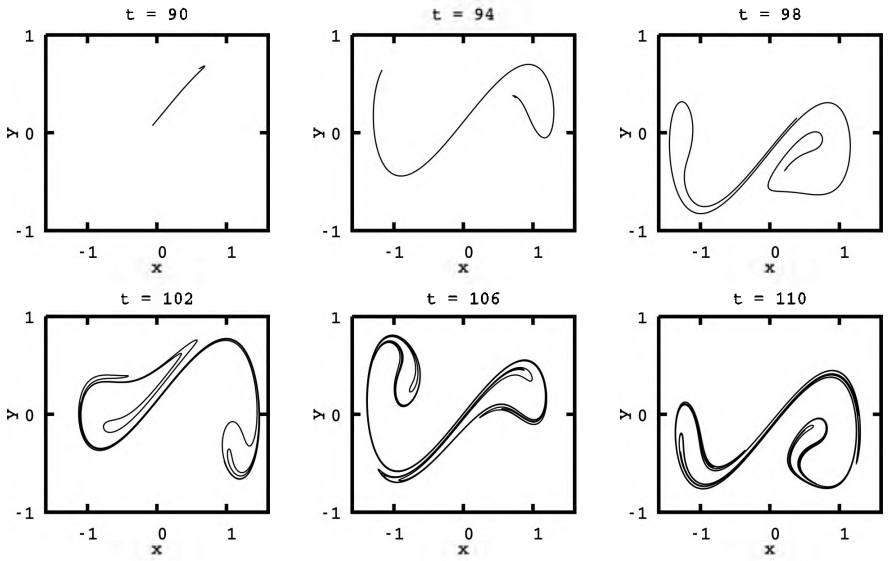

Рис. 65. Множество решений системы (4.2) в различные моменты времени ($t \in [90, 110]$)

Вначале рассмотрим систему ОДУ с одним малым интервальным начальным условием:

$$\begin{cases} x' = y, \\ y' = x - 0.25y - x^3 + 0.3\cos(t), \\ x(0) \in 1 + \left[-10^{-3}, 10^{-3}\right],\ y(0) = 1, \\ t \in [0, 110]. \end{cases} \tag{4.2}$$

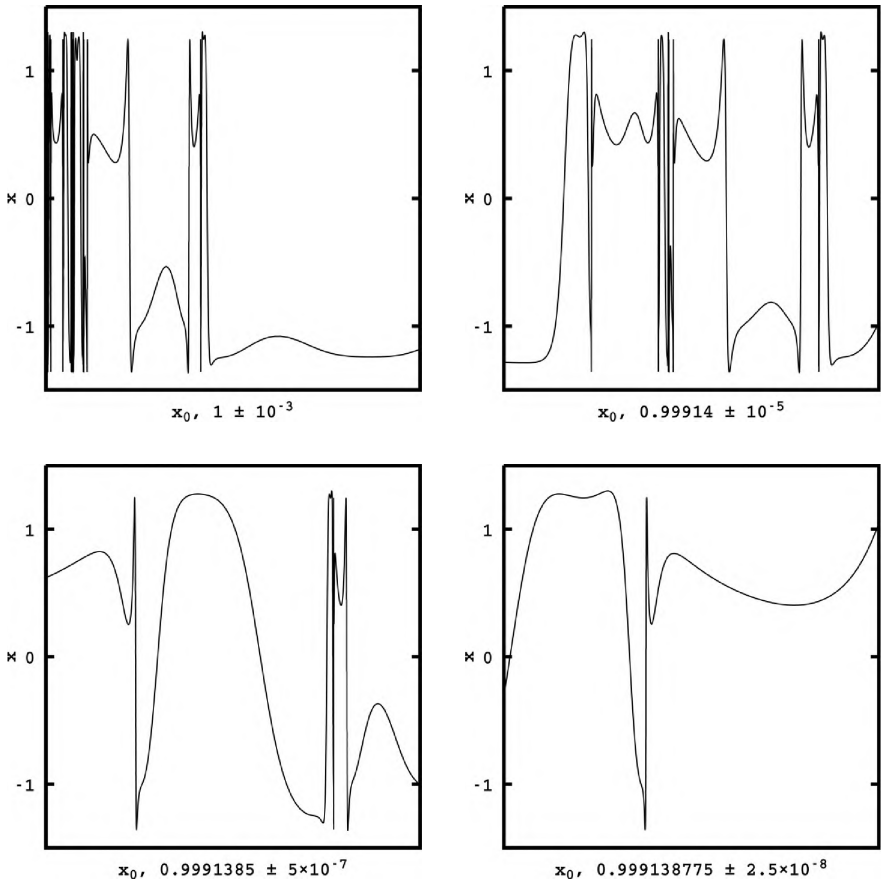

Рис. 66. Структура множества решений системы (4.2) в конечный момент времени

На рис. 64 и 65 показано множество решений в различные моменты времени (шаг по времени — 0.1). Вначале множество представляет собой короткий отрезок, практически точку. Постепенно эта точка вытягивается в линию. После момента $t = 90$ (см. рис. 65) кривая начинает затейливым образом запутываться, извиваться и растягиваться еще сильнее, что характерно для хаоса.

На рис. 66 показана зависимость решения от интервального параметра в момент времени $t = 110$. При просмотре слева направо и сверху вниз каждая картинка является отмасштабированной частью предыдущей картинки. Здесь наблюдаются элементы самоподобия, т.е. элементы фрактальной структуры решения.

Рассмотрим задачу Коши с двумя «большими» интервальными начальными условиями [47]:

$$\begin{cases} x' = y, \\ y' = x - 0.25y - x^3 + 0.3\cos(t), \\ x(0) \in [-2, 2],\ y(0) \in [-2, 2], \\ t \in [0, 7]. \end{cases} \tag{4.3}$$

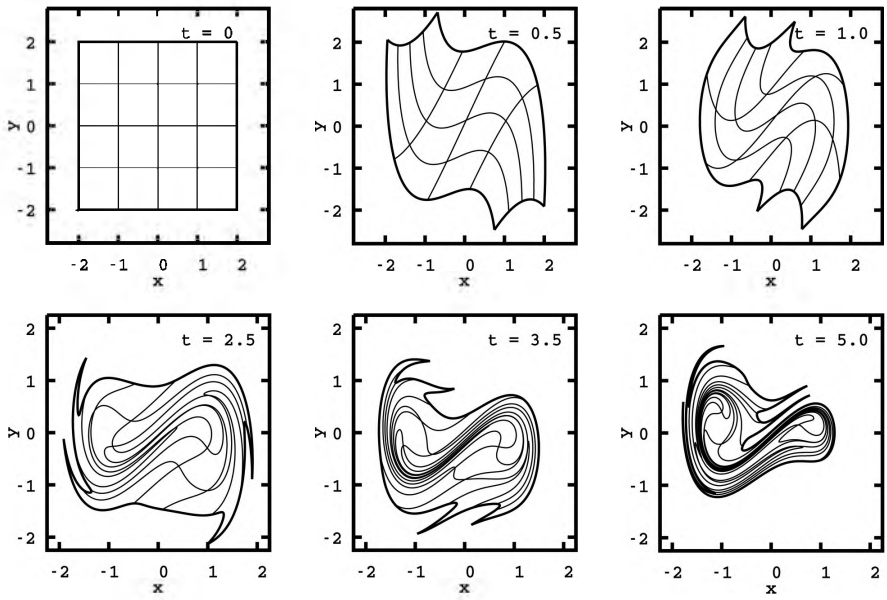

Рис. 67. Множество решений системы (4.3) в различные моменты времени

На рис. 67 и 68 (верхний график) показано множество решений в различные моменты времени. В начальный момент решение является квадратом. В процессе интегрирования структура множества сильно меняется, оно скручивается и уменьшается в размерах. На рис. 69 показаны получающиеся сетки. Здесь наблюдается уплотнение ячеек.

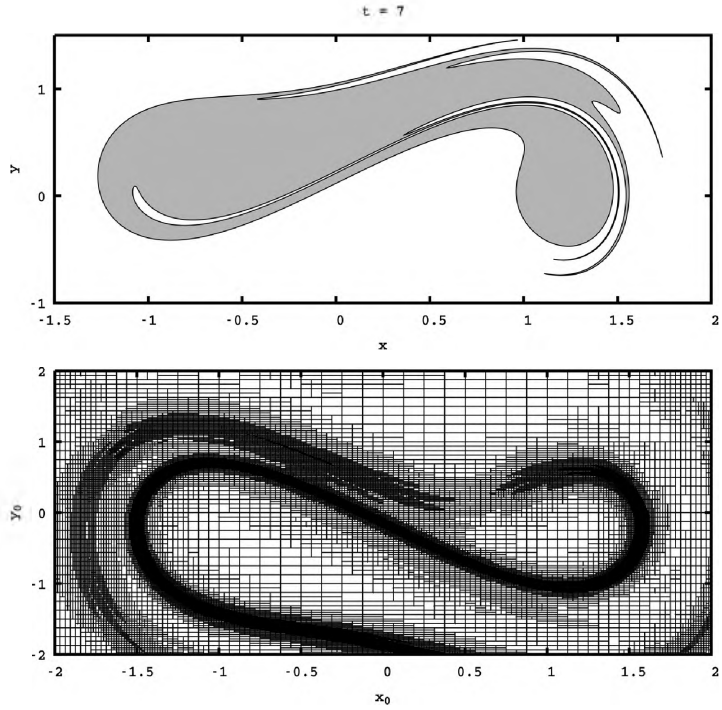

Рис. 68. Множество решений и разбиение пространства в конечный момент времени

На рис. 70 и 71 представлены зависимости решений от начальных условий. Увеличение четкости со временем свидетельствует о зарождении хаоса и возникновении резких изменений в решениях. Граница между темными и светлыми областями говорит о том, что решения, изначально находившиеся близко друг к другу, в итоге оказываются далеко. Чем ярче выделяется эта граница, тем больше требуется уплотнение сетки в ее окрестности (рис. 68, нижний график). В целом, данная картина является обобщением рис. 62: чем плотнее сетка, тем больше глубина соответствующих вершин в дереве.

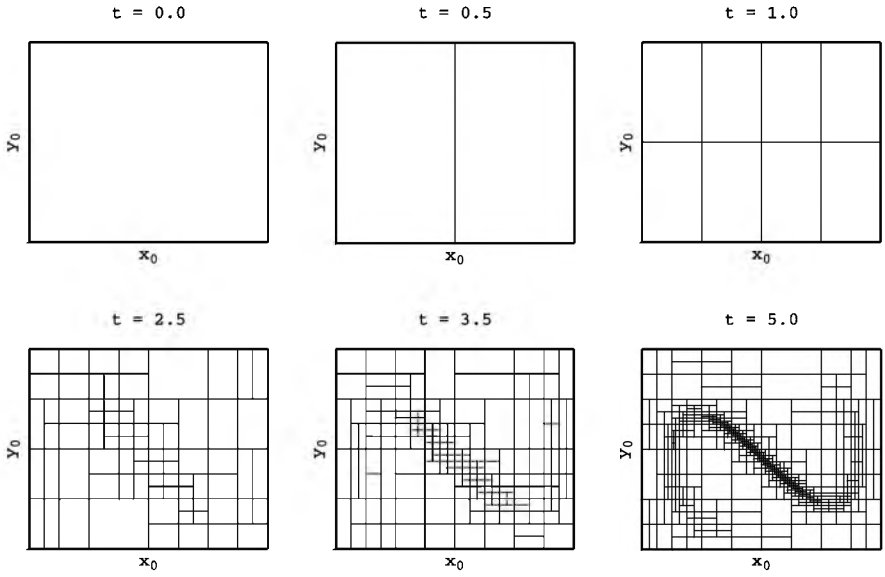

Рис. 69. Разбиения пространства, получающиеся в процессе решения системы (4.3)

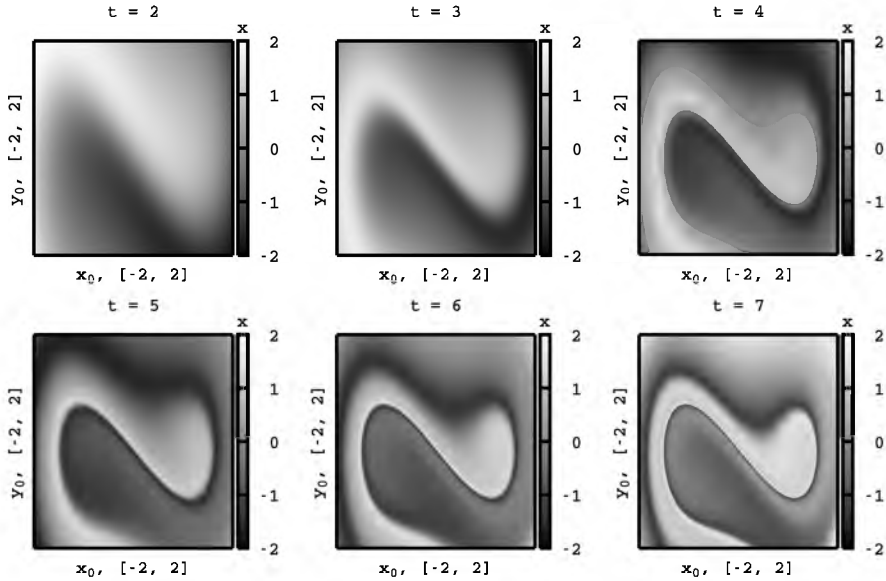

Рис. 70. Зависимость x от интервальных начальных условий системы (4.3)

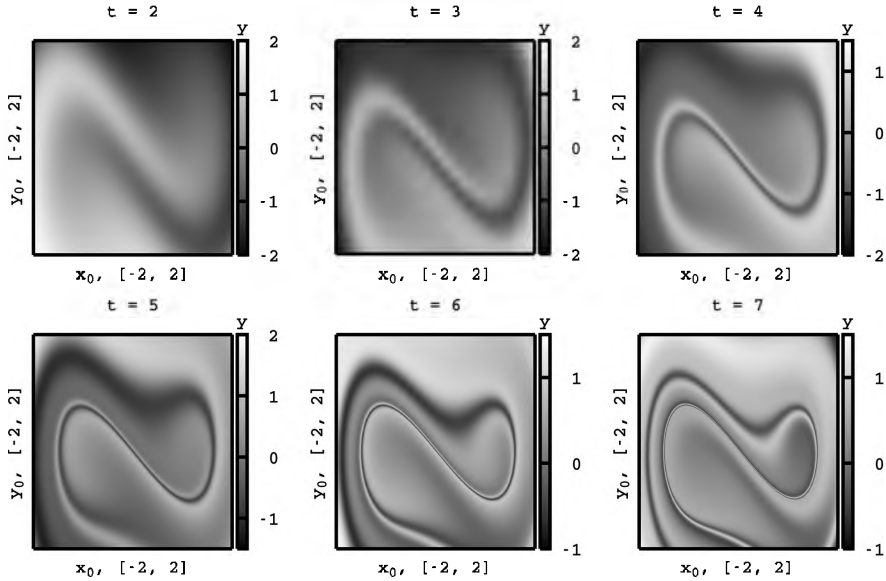

Рис. 71. Зависимость y от интервальных начальных условий системы (4.3)

Появление выраженного уплотнения ячеек, которое со временем только растет, не перемещаясь в пространстве, является признаком неустойчивости и хаоса.

4.1.3. Аттрактор Лоренца

В 1961 году Эдвард Лоренц из Массачусетского технологического института занимался численным исследованием метеосистем — в частности, моделированием конвекционных токов в атмосфере. Он численно решал следующую систему уравнений:

$$\begin{cases} x' = \sigma(-x + y), \\ y' = rx - y - xz, \\ z' = -bz + xy, \end{cases}$$

где $\sigma = -10$, $r = 28$ и $b = 8/3$. В процессе проведения вычислений Лоренц в определенный момент уменьшил число верных десятичных знаков в промежуточной точке интегрирования, в результате получил

решение, совершенно отличающееся от ранее полученного решения. То, что он обнаружил, называется теперь «существенной зависимостью от начальных условий» (или эффектом бабочки) и является основной чертой, присущей хаотической динамике [99].

Подобные системы являются хорошими тестами для интервальных методов. Так, каким бы малым ни был исходный интервал, все равно в нем найдутся две точки, которым соответствуют кардинально разные решения. Если говорить нестрогим языком, то в пределе можно считать, что решение со временем становится «разрывным» по начальному условию. После определенного момента времени абсолютно любой метод аварийно завершит расчет. Алгоритм адаптивной интерполяции не является исключением. В процессе вычислений будет происходить постоянное уплотнение сетки, в результате чего в конечном счете может не хватить вычислительных ресурсов. Для того чтобы получить информацию о структуре решения на максимально длинном интервале интегрирования, вводится минимальный размер ячейки. Ячейки, размер которых меньше минимального размера, помечаются как «плохие», а соответствующая область неопределенности считается «областью разрыва».

Сначала рассмотрим задачу Коши, содержащую небольшой интервал:

$$\begin{cases} x' = \sigma(-x + y), \\ y' = rx - y - xz, \\ z' = -bz + xy, \\ x(0) = 1,\ y(0) = 0,\ z(0) \in \left[-10^{-3}, 10^{-3}\right], \\ t \in [0, 27]. \end{cases} \tag{4.4}$$

На рис. 72 показано множество решений в различные моменты времени (шаг по времени — 0.05). Сначала это множество представляет собой небольшой отрезок, параллельный оси z, который на первый взгляд ничем не отличается от точки. Постепенно в процессе вычислений он вытягивается и уже становится различим на фазовой плоскости ($t = 20$). Спустя еще некоторое время (рис. 73) он растягивается настолько, что заполняет практически все фазовое

пространство. Его длина увеличивается экспоненциально, и начинает проявляться эффект бабочки.

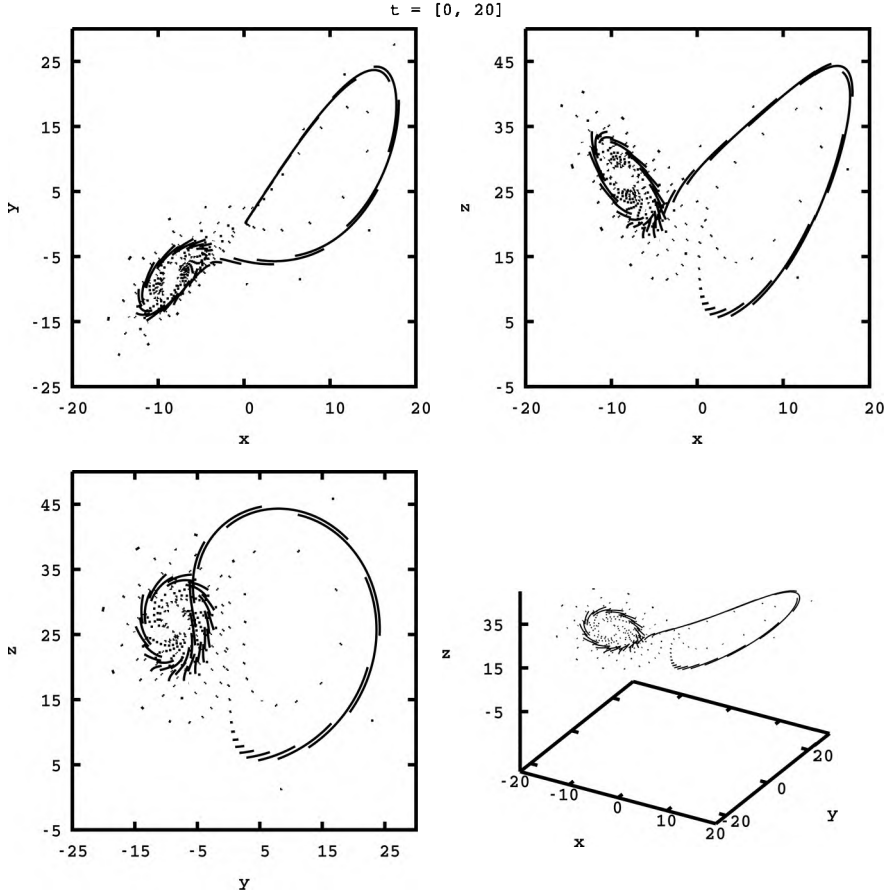

Рис. 72. Множество решений системы (4.4) в различные моменты времени ($t \in [0, 20]$)

На рис. 74 показана зависимость x от интервального начального условия в момент времени $t = 27$. При просмотре слева направо и сверху вниз каждая картинка является отмасштабированной частью предыдущей картинки. Чтобы рассмотреть все детали структуры решения, необходимо изменить масштаб на четыре порядка. При

этом со временем здесь наблюдается экспоненциальное усложнение картины. Пунктирные линии соответствуют «плохим» ячейкам. При вычислении решений, которые попадают в них, используется линейная интерполяция.

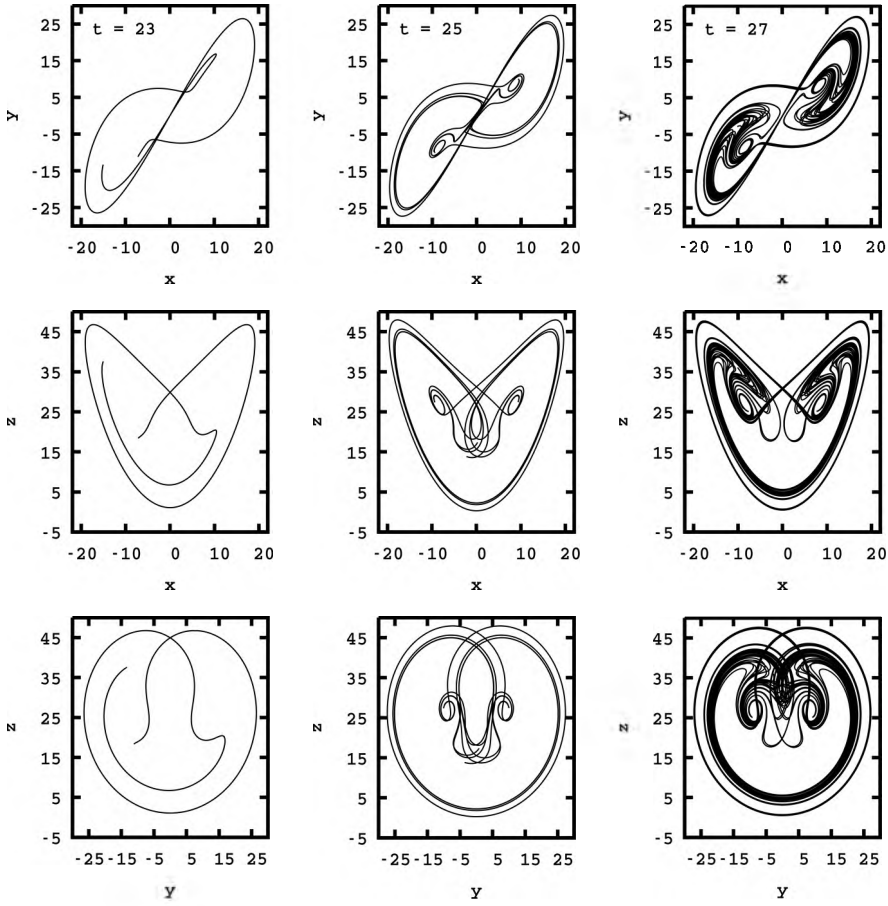

Рис. 73. Множество решений системы (4.4) в различные моменты времени ($t \in [23, 27]$)

Далее рассмотрим систему ОДУ, содержащую два интервальных параметра:

$$\begin{cases} x' = \sigma(-x + y), \\ y' = rx - y - xz, \\ z' = -bz + xy, \\ x(0) = 1,\ y(0) = 0,\ z(0) = 0, \\ t \in [0, 19], \end{cases} \tag{4.5}$$

где $\sigma \in [10, 11]$ и $r \in [28, 29]$.

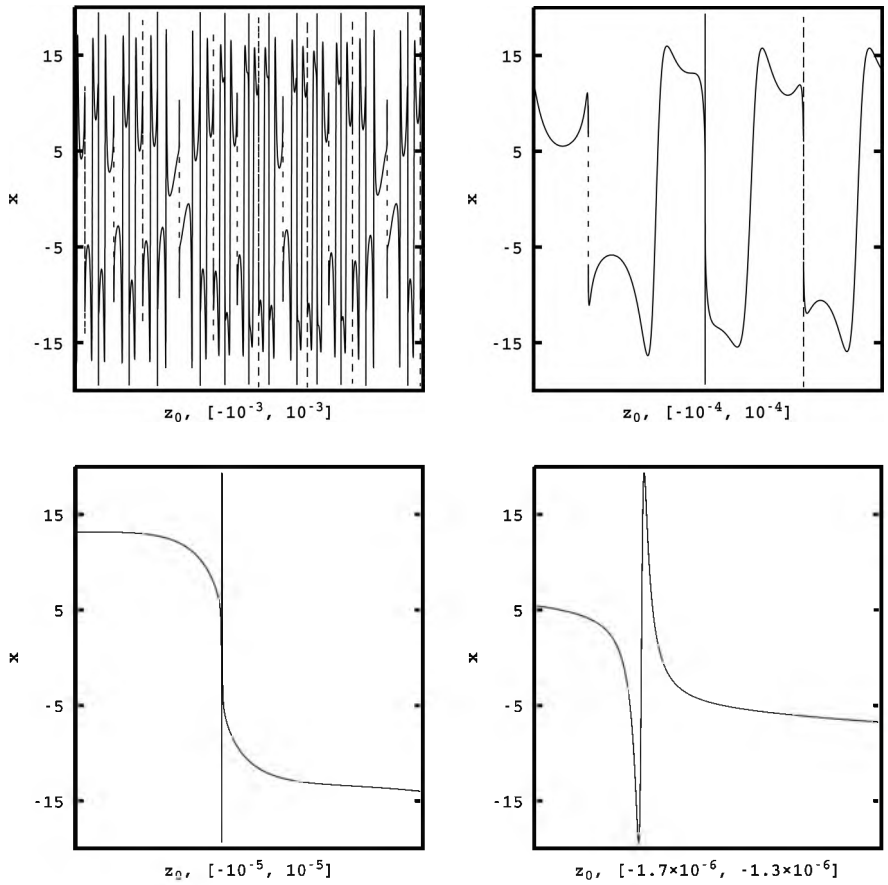

Рис. 74. Структура множества решения системы (4.4) в конечный момент времени

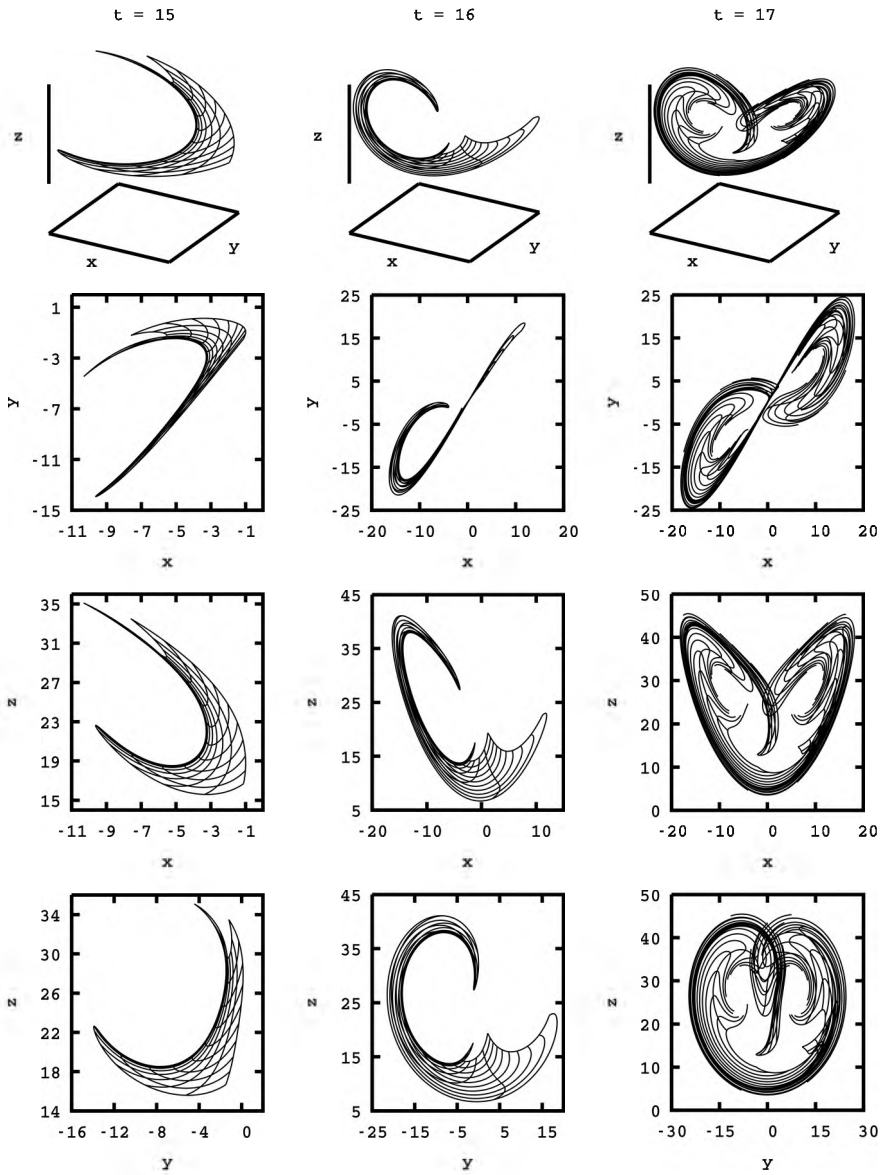

Рис. 75. Множество решений системы (4.5) в различные моменты времени

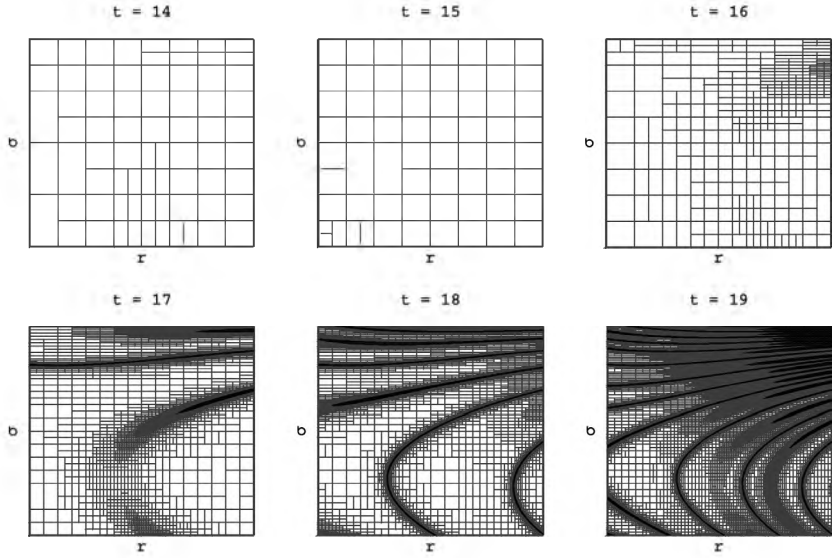

Рис. 76. Разбиения пространства, получающиеся в процессе решения системы (4.5)

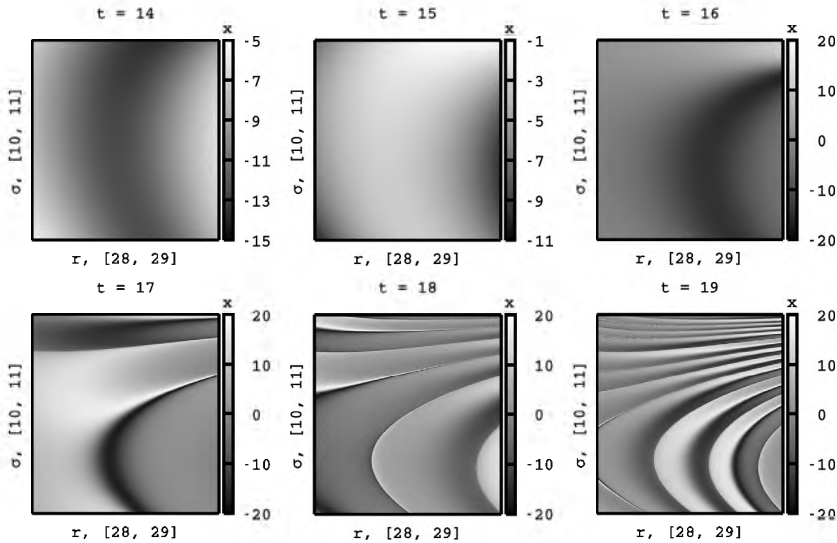

Рис. 77. Зависимость x от интервальных параметров системы (4.5)

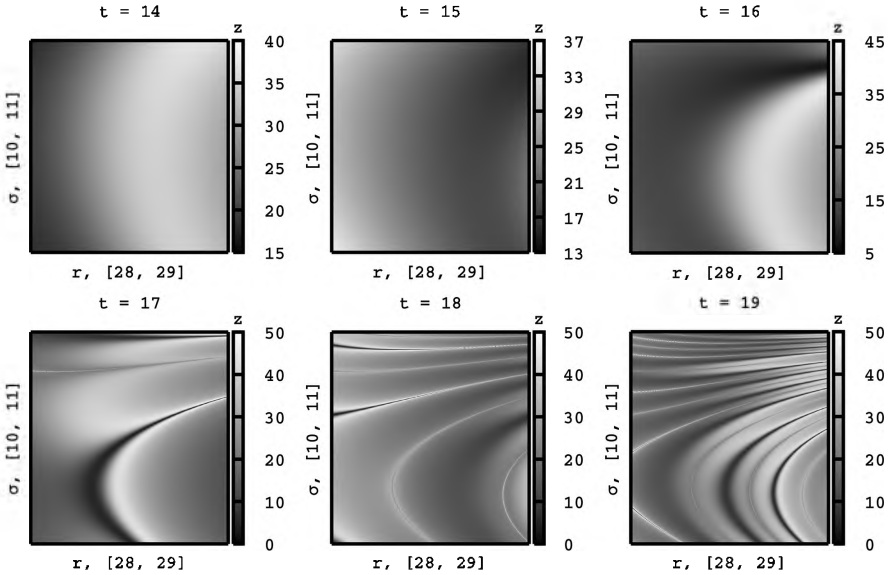

Рис. 78. Зависимость z от интервальных параметров системы (4.5)

На рис. 75 показаны проекции множества решений системы (4.5) в различные моменты времени. На проекциях (момент времени $t = 17$, правый столбик) видно, что некоторые линии обрываются и не замыкаются. Это связано с тем, что в процессе интегрирования начинают возникать «плохие» ячейки и соответствующие им решения не отображаются. На рис. 76 представлена получающаяся сетка в процессе вычислений, где черным цветом выделены «плохие» ячейки. С течением времени здесь наблюдается экспоненциальное увеличение количества линий разрыва. На рис. 77 и 78 отражены зависимости решений от интервальных параметров. Как и в случае с осциллятором Дуффинга (см. рис. 70, 71), увеличение четкости картины и появление резких переходов цветов свидетельствует о том, что в системе имеется существенная зависимость от параметров, то есть незначительное изменение параметров в правой части системы ОДУ приводит к возникновению принципиально разных решений.

4.2. АСТЕРОИДНАЯ ОПАСНОСТЬ

До июля 1994 года разговоры об опасности столкновения нашей планеты с астероидом, кометой или крупным метеоритом многими учеными воспринимались как не совсем научная фантастика, но столкновение кометы Шумейкеров–Леви с Юпитером, хорошо видимое в телескоп, заставило изменить мнение научного сообщества на противоположное. Обломки кометы, образовавшиеся во время предыдущего прохождения вблизи Юпитера, врезались по очереди в атмосферу Юпитера со скоростью 40–60 километров в секунду. Больше десяти лет астрономы Земли наблюдали возмущения в атмосфере газового гиганта — последствия этого падения. Для устойчивого существования нашей цивилизации жители планеты Земля должны иметь защиту от подобных столкновений. Для этого необходимо:

во-первых, вовремя обнаружить опасные объекты, точно измерить их координаты, скорость, массу и размеры;

во-вторых, рассчитать закон движения до возможного столкновения с Землей [100, 101];

в-третьих, выбрать оптимальный метод изменения траектории опасного объекта, чтобы он пролетел мимо нашей планеты.

При этом уже на первом этапе возникают сложности. Определить точно параметры движения астероида невозможно, поэтому возникают неопределенности, которые могут быть представлены как интервалы.

Рассмотрим движение астероида вокруг Солнца [43] без учета влияния других планет (рис. 79). Такая задача, в которой рассматривается движение только двух масс под действием гравитационной силы, называется проблемой Кеплера. Поскольку масса астероида во много раз меньше массы Солнца, то можно считать, что Солнце стоит на месте, а астероид движется вокруг него. Здесь траектория, по которой будет двигаться астероид, описывается дифференциальным уравнением

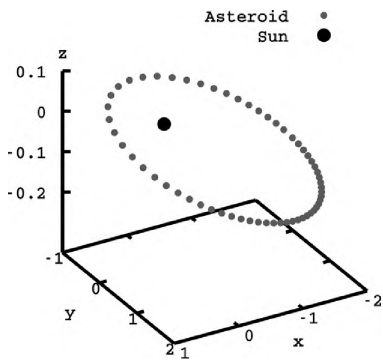

Рис. 79. Движение астероида вокруг Солнца

$$\vec{r}'' = -\gamma \frac{\vec{r}}{\left(r_x^2 + r_y^2 + r_z^2\right)^{3/2}},$$

где $\vec{r}(t) = (r_x(t), r_y(t), r_z(t))$ — координаты астероида относительно положения Солнца, $\gamma = 0.9986$ — константа, получаемая произведением гравитационной постоянной G на массу Солнца. Единицы измерения — астрономические единицы. Время отмасштабировано таким образом, что один земной год равняется 2π.

Задача Коши для описания такого процесса имеет вид:

$$\begin{cases} x' = v_x,\ y' = v_y,\ z' = v_z, \\ v_x' = -\gamma \dfrac{x}{d},\ v_y' = -\gamma \dfrac{y}{d},\ v_z' = -\gamma \dfrac{z}{d}, \\ d = \left(x^2 + y^2 + z^2\right)^{3/2}, \\ x(0) \in -1.77269098191512 + \left[-5 \times 10^{-8},\ 5 \times 10^{-8}\right], \\ y(0) \in 0.1487214852342955 + \left[-5 \times 10^{-8},\ 5 \times 10^{-8}\right], \\ z(0) \in -0.07928350462244194 + \left[-5 \times 10^{-8},\ 5 \times 10^{-8}\right], \\ v_x(0) \in 0.2372031916516237 + \left[-5 \times 10^{-7},\ 5 \times 10^{-7}\right], \\ v_y(0) \in -0.612524538758628 + \left[-5 \times 10^{-7},\ 5 \times 10^{-7}\right], \\ v_z(0) \in 0.04583217572165624 + \left[-5 \times 10^{-7},\ 5 \times 10^{-7}\right], \\ t \in [0,\ 46\pi]. \end{cases} \tag{4.6}$$

Начальные условия соответствуют измеренному положению и скорости астероида XF11 на 17 января 1997 года. Диаметр интервалов включает в себя все погрешности измерения.

Далее выполним интегрирование системы (4.6) и сравним решение с решениями, полученными разными библиотеками в работах [43, 49]. Расчеты с помощью COSY-VI, AWA и RiOT выполнялись на процессоре AMD Sempron 2600+, verifyode — на процессоре Intel Xeon E5-1620. Библиотека verifyode является решателем интервальных систем ОДУ. Она реализована в MATLAB/INTLAB и включает в себя модель Тейлора с механизмом Shrink Wrapping.

Таблица 4.1

Сравнение результатов решения системы (4.6) в момент времени 5.5π

	Точное решение	Алгоритм адаптивной интерполяции	COSY-VI
x	[–0.5671422, – 0.5670997]	[**–0.5671422**, – **0.5670997**]	[**–0.5671**453, – **0.5670**966]
y	[1.8387331, 1.8387380]	[**1.8387331**, **1.8387380**]	[**1.838732**, **1.838739**]
z	[–0.1318261, – 0.1318215]	[**–0.1318261**, – **0.1318215**]	[**–0.131826**3, – **0.131821**4]
$\mathbf{v_x}$	[–0.5867546, – 0.5867509]	[**–0.5867546**, – **0.5867509**]	[**–0.586755**0, – **0.586750**6]
$\mathbf{v_y}$	[0.0499695, 0.0499870]	[**0.0499695**, **0.0499870**]	[**0.049968**1, **0.049988**5]
$\mathbf{v_z}$	[–0.0262877, – 0.0262865]	[**–0.026287**7, – **0.0262865**]	[**–0.026287**9, – 0.**026286**4]
время, с	—	0.591	1702
	AWA	RiOT	verifyode
x	[**–0.567143**9, – **0.567098**0]	[**–0.567141**3, – **0.567**1006]	[**–0.567142**7, – **0.567099**2]
y	[**1.8387331**, **1.838740**]	[**1.838733**, **1.838739**]	[**1.838733**, **1.838739**]
z	[**–0.131826**3, – **0.131821**3]	[**–0.1318261**, – **0.131821**6]	[**–0.131826**3, – **0.131821**3]
$\mathbf{v_x}$	[**–0.586755**2, – **0.586750**3]	[**–0.586754**5, – **0.586751**1]	[**–0.586754**7, – **0.5867509**]
$\mathbf{v_y}$	[**0.049968**3, **0.049988**1]	[**0.049969**6, **0.049986**8]	[**0.049969**0, **0.049987**5]
$\mathbf{v_z}$	[**–0.026287**8, – **0.0262865**]	[**–0.0262877**, – **0.026286**5]	[**–0.026287**8, – **0.0262865**]
время, с	3.01	34336	3401

Таблица 4.2

Сравнение результатов решения системы (4.6) в момент времени 11π

	Точное решение	Алгоритм адаптивной интерполяции	COSY-VI
x	[–0.9188592, – 0.9187385]	[**–0.9188592**, – **0.9187385**]	[**–0.9188**739, – **0.9187**245]
y	[–0.7430433, – 0.7429904]	[**–0.7430433**, – **0.7429904**]	[**–0.7430**500, – **0.7429**834]
z	[0.0076459, 0.0076547]	[**0.0076459**, **0.0076547**]	[**0.0076**449, **0.0076**556]
$\mathbf{v_x}$	[0.9136817, 0.9137549]	[**0.9136817**, **0.9137549**]	[**0.9136**730, **0.9137**633]
$\mathbf{v_y}$	[–0.4045054, – 0.4044449]	[**–0.4045054**, – **0.4044448**]	[**–0.4045**129, – **0.4044**378]
$\mathbf{v_z}$	[0.0603496, 0.0603508]	[**0.0603496**, **0.0603508**]	[**0.060349**5, **0.060350**9]
время, с	—	1.21	3427
	AWA	RiOT	verifyode
x	[**–0.91**92449, – **0.918**3529]	[**–0.9**231145, – **0.91**44781]	[**–0.918859**6, – **0.918738**2]
y	[**–0.743**2376, – **0.742**7962]	[**–0.74**49016, – **0.74**11222]	[**–0.743043**9, – **0.7429**899]
z	[**0.0076**204, **0.0076**802]	[**0.007**3390, **0.007**9610]	[**0.0076459**, **0.0076547**]
$\mathbf{v_x}$	[**0.913**4469, **0.913**9898]	[**0.91**12709, **0.91**61645]	[**0.913681**5, **0.913755**2]
$\mathbf{v_y}$	[**–0.404**7165, – **0.404**2339]	[**–0.40**63907, – **0.40**15455]	[**–0.4045057**, – **0.4044447**]
$\mathbf{v_z}$	[**0.0603456**, **0.0603547**]	[**0.0603**196, **0.0603**798]	[**0.0603496**, **0.0603508**]
время, с	5.26	111640	7 466

Таблица 4.3

Сравнение результатов решения системы (4.6) в момент времени 46π

	Точное решение	Алгоритм адаптивной интерполяции	COSY-VI
x	[–0.9037235, – 0.9031897]	[**–0.9037233**, – **0.9031895**]	[**–0.903**9513, – **0.90**29650]
y	[–0.7498251, – 0.7495961]	[**–0.7498253**, – **0.7495963**]	[**–0.749**9257, – **0.749**4944]
z	[0.0086410, 0.0086746]	[**0.0086410**, **0.0086747**]	[**0.0086**243, **0.0086**912]
$\mathbf{v_x}$	[0.9228591, 0.9231813]	[**0.922859**2, **0.923181**4]	[**0.922**7239, **0.923**3148]
$\mathbf{v_y}$	[–0.3969893, – 0.3967206]	[**–0.396989**2, – **0.396720**5]	[**–0.39**71035, – **0.396**6078]
$\mathbf{v_z}$	[0.0602651, 0.0602688]	[**0.0602651**, **0.0602688**]	[**0.060263**5, **0.060270**3]
время, с	—	4.47	16316

Для алгоритма адаптивной интерполяции задавался порядок $p = 2$, шаг интегрирования неинтервальных ОДУ — $h = 10^{-2}$ и относительная погрешность — 10^{-5}. Вычисления производились на процессоре Intel(R) Core(TM) i5-4200M CPU @ 2.50GHz.

В табл. 4.1 и 4.2 приводятся решения, полученные различными программными библиотеками в моменты времени 5.5π (2.75 года) и 11π (5.5 года) соответственно. Все рассмотренные библиотеки, за исключением COSY-VI (табл. 4.3), аварийно завершают расчет при $t > 15\pi$ (7.5 года). При этом требуемое вычислительное время измеряется часами, а иногда и сутками (табл. 4.2, библиотека RiOT). Даже если принять во внимание, что расчеты выполнялись на достаточно разнородных процессорах, алгоритм адаптивной интерполяции справляется с рассмотренной задачей на порядки быстрее и лучше в плане получаемых интервальных оценок.

4.3. ХИМИЧЕСКАЯ КИНЕТИКА И ГАЗОВАЯ ДИНАМИКА

Для моделирования газофазных химических превращений необходимо знать кинетический механизм и скорости протекающих реакций. Как правило, зависимости, которые описывают скорости, получают экспериментальными способами, зачастую дающими лишь приближенные значения [102]. Далее будет показано, что значения функций, аппроксимирующих скорость протекания одной и той же реакции, но полученных разными исследователями, могут различаться в десятки и в сотни раз. Естественно, для численного моделирования в этом случае необходимо применение интервальных методов, которые могут работать в условиях больших неопределенностей.

Многокомпонентная система переменного состава из N веществ, в которой протекает N_r реакций, имеет вид [103]:

$$\sum_{i=1}^{N} \vec{v}_i^{(r)} M_i \underset{\overline{W}^{(r)}}{\overset{\overline{W}^{(r)}}{\rightleftarrows}} \sum_{i=1}^{N} \overleftarrow{v}_i^{(r)} M_i,\ \vec{q}^{(r)} = \sum_{i=1}^{N} \vec{v}_i^{(r)},\ r = \overline{1, N_r},$$

где r — порядковый номер реакции; $\overleftrightarrow{v}_i^{(r)}$ — стехиометрические коэффициенты; $\vec{q}^{(r)}$ — молекулярность элементарных реакций; M_i — сим-

волы молекул или атомов химических компонентов; $\overleftrightarrow{W}^{(r)}$ — скорости r-й реакции в прямом и обратном направлении. Последняя величина определяет изменение концентрации компонентов с течением времени в единице объема смеси.

Скорость химической реакции $\overleftrightarrow{W}^{(r)}$ определяется как произведение объемных концентраций компонентов и «константы» скорости реакции $\overleftrightarrow{K}^{(r)}(T)$, зависящей от температуры:

$$\overleftrightarrow{W}^{(r)} = \overleftrightarrow{K}^{(r)}(T)\prod_i (\rho\gamma_i)^{\overleftrightarrow{v}_i^{(r)}},$$

где ρ — плотность; γ_i — мольно-массовая концентрация i-го компонента.

Для аппроксимации температурной зависимости констант скоростей прямых реакций используется обобщенная формула Аррениуса:

$$\overrightarrow{K}(T) = AT^n \exp\left(-\frac{E}{T}\right),$$

где A, n, E — некоторые постоянные величины, индивидуальные для каждой реакции. Именно в этих величинах и содержится неопределенность.

Константы скоростей обратных реакций вычисляются исходя из константы равновесия:

$$\overleftarrow{K}^{(r)}(T) = \overrightarrow{K}^{(r)}(T)\exp\left[\sum_i^N \left(\overleftarrow{v}_i^{(r)} - \overrightarrow{v}_i^{(r)}\right)\left(\frac{G_i^0(T)}{RT} + \ln\frac{RT}{P_0}\right)\right],$$

где $P_0 = 101325$ Па — стандартное давление; R — универсальная газовая постоянная; $G_i^0(T)$ — стандартный молярный потенциал Гиббса i-го компонента, который задается с помощью полиномов из справочника [104].

Скорость образования i-го компонента имеет следующий вид:

$$W_i = \sum_{r=1}^{N_r}\left(\overleftarrow{v}_i^{(r)} - \overrightarrow{v}_i^{(r)}\right)\left(\overleftarrow{W}^{(r)} - \overrightarrow{W}^{(r)}\right), i = \overline{1, N}.$$

Все термодинамические величины для смеси идеальных газов выражаются через стандартный молярный потенциал Гиббса. Далее приводятся некоторые основные соотношения.

Удельный потенциал Гиббса:

$$G(T,\ P,\ \bar{\gamma}) = \sum_{i=1}^{N} \gamma_i \left[G_i^0(T) + RT \ln \left(\frac{P\gamma_i}{P_0 \sum_{j=1}^{N} \gamma_j} \right) \right].$$

Энтропия:

$$S(T,\ P,\ \bar{\gamma}) = \sum_{i=1}^{N} \gamma_i \left[-\frac{\partial G_i^0(T)}{\partial T} - R \ln \left(\frac{P\gamma_i}{P_0 \sum_{j=1}^{N} \gamma_j} \right) \right].$$

Энтальпия (калорическое уравнение):

$$H(T,\ \bar{\gamma}) = \sum_{i=1}^{N} \gamma_i \left(G_i^0(T) - T \frac{\partial G_i^0(T)}{\partial T} \right).$$

Внутренняя энергия (калорическое уравнение):

$$U(T,\ \bar{\gamma}) = H(T,\ \bar{\gamma}) - RT \sum_{j=1}^{N} \gamma_i \ .$$

Изобарная теплоемкость:

$$C_p(T,\ \bar{\gamma}) = \frac{\partial H(T,\ \bar{\gamma})}{\partial T}.$$

Молярная теплоемкость:

$$C_v(T,\ \bar{\gamma}) = C_v(T,\ \bar{\gamma}) - R \sum_{i=1}^{N} \gamma_i \ .$$

Показатель адиабаты:

$$\kappa(T,\ \bar{\gamma}) = \frac{C_p(T,\ \bar{\gamma})}{C_v(T,\ \bar{\gamma})}.$$

Скорость звука:

$$a(T,\ \bar{\gamma}) = \sqrt{\kappa(T,\ \bar{\gamma}) RT \sum_{i=1}^{N} \gamma_i} \ .$$

Уравнение состояния смеси идеальных газов (термическое уравнение):

$$PV = RT\sum_{i=1}^{N}\gamma_i \rightarrow P = \rho RT\sum_{i=1}^{N}\gamma_i \ .$$

Для демонстрации неопределенностей, содержащихся в константах скоростей химических реакций, рассмотрим кинетические механизмы, приведенные в работах [105] (табл. 4.4) и [106] (табл. 4.5). Большая часть содержащихся в таблицах реакций не совпадает по направлению, поэтому для их сравнения воспользуемся константой равновесия. На рис. 80 показаны зависимости констант скоростей от температуры для разных механизмов. Наиболее сильное несовпадение кривых наблюдается для реакций 1, 3 и 8 из механизма 2. Учтем это расхождение в соответствующих коэффициентах (табл. 4.6).

Таблица 4.4

Первый механизм горения смеси H_2-O_2

Реакция	A, м, моль, с	n	E, К
1. $O_2+H \rightarrow OH+O$	2.0×10^{8}	0.0	8455
2. $H_2+O \rightarrow OH+H$	5.06×10^{-2}	2.67	3163
3. $H_2+OH \rightarrow H_2O+H$	1.0×10^{2}	1.6	1659
4. $OH+OH \rightarrow H_2O+O$	1.5×10^{3}	1.14	50
5. $H+H+M \rightarrow H_2+M$	1.8×10^{6}	–1.0	0
6. $O+O+M \rightarrow O_2+M$	2.9×10^{5}	–1.0	0
7. $H+OH+M \rightarrow H_2O+M$	2.2×10^{10}	–2.0	0

Таблица 4.5

Второй механизм горения смеси H_2-O_2

Реакция	A, м, моль, с	n	E, К
1. $H_2O+H \rightarrow OH+H_2$	8.4×10^{7}	0	10116
2. $O_2+H \rightarrow OH+O$	2.2×10^{8}	0	8455
3. $H_2+O \rightarrow OH+H$	1.8×10^{4}	1	4480
4. $O_2+M \rightarrow 2O+M$	5.4×10^{12}	–1	59400
5. $H_2+M \rightarrow 2H+M$	2.2×10^{8}	0	48300
6. $H_2O+M \rightarrow OH+H+M$	10^{18}	–2.2	59000
7. $HO+M \rightarrow O+H+M$	8.5×10^{12}	–1	50830
8. $H_2O+O \rightarrow 2OH$	5.8×10^{7}	0	9059

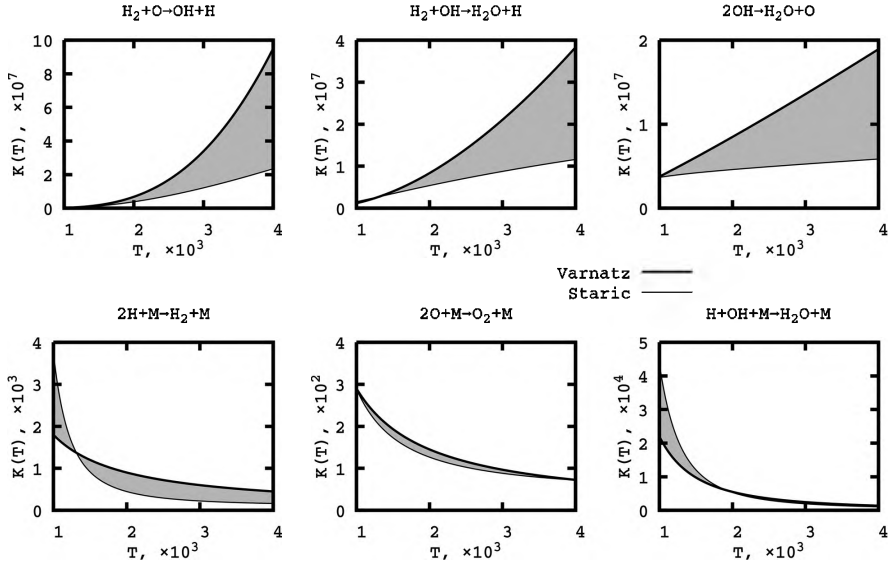

Рис. 80. Сравнение различных механизмов горения смеси H_2-O_2

Таблица 4.6

Интервальная часть механизма горения смеси H_2-O_2

Реакция	A, м, моль, с	n	E, K
1. $H_2O+H \rightarrow OH+H_2$	$[8.4\times10^7, 4.2\times10^8]$	0	10116
3. $H_2+O \rightarrow OH+H$	$[1.8\times10^4, 9\times10^4]$	1	4480
8. $H_2O+O \rightarrow 2OH$	$[5.8\times10^7, 2.9\times10^8]$	0	9059

4.3.1. Модель адиабатической реакции

Рассматривается стехиометрическая смесь водорода и кислорода при начальной температуре $T = 1200$ K, постоянной плотности $\rho = 0.122$ кг/м^3 и постоянной внутренней энергии $U = 1.48$ МДж/кг. Модель химической кинетики задается системой из шести компонентов (H_2O, OH, H_2, O_2, H, O), в которой протекает восемь реакций (табл. 4.5 и 4.6). Здесь система ОДУ записывается следующим образом:

$$\frac{d\gamma_i}{dt} = \frac{1}{\rho}\sum_{r=1}^{N_r}\left(\bar{\nu}_i^{(r)} - \bar{\nu}_i^{(r)}\right)\left(\overline{W}^{(r)} - \overline{W}^{(r)}\right),\ i = \overline{1,\ N}$$

и дополняется уравнением сохранения внутренней энергии системы:

$$U(T, \bar{\gamma}) = H(T, \bar{\gamma}) - RT \sum_{j=1}^{N} \gamma_i .$$

При каждом вычислении правой части системы ОДУ уравнение внутренней энергии разрешается относительно T с помощью метода Ньютона.

Начальные условия:

$$\gamma_{H_2} = 55.50868,$$
$$\gamma_{O_2} = 27.75434,$$
$$\gamma_{H_2O} = \gamma_{OH} = \gamma_H = \gamma_O = 0.$$

Полученная система ОДУ является жесткой, и для ее интегрирования используется неявный метод Розенброка с заморозкой матрицы Якоби [107]. На рис. 81 представлены верхние и нижние оценки для концентраций всех компонентов смеси. Неопределенность в константах скоростей реакций приводит к неопределенности важного параметра — времени задержки воспламенения в диапазоне от 22 до 29 мкс.

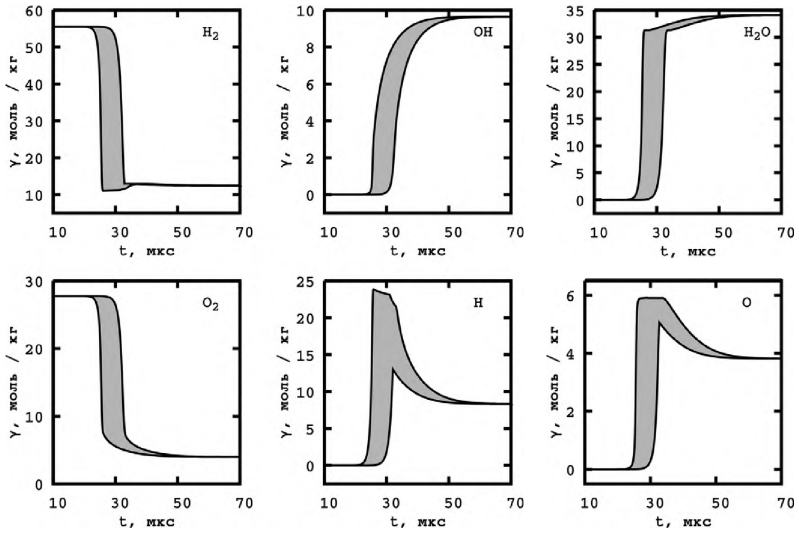

Рис. 81. Зависимость мольно-массовых концентраций H_2, OH, H_2O, O_2, H и O от времени

Отметим, что для данной модели характерным является переход в равновесное состояние, которое никак не зависит от констант скоростей реакций, о чем свидетельствует совпадение верхних оценок концентраций с нижними оценками после определенного момента времени. Такая особенность отражается в динамике перестроения kd-дерева (рис. 82 и 83). После совпадения решений дерево полностью «схлопывается» и от него остается только одна корневая вершина.

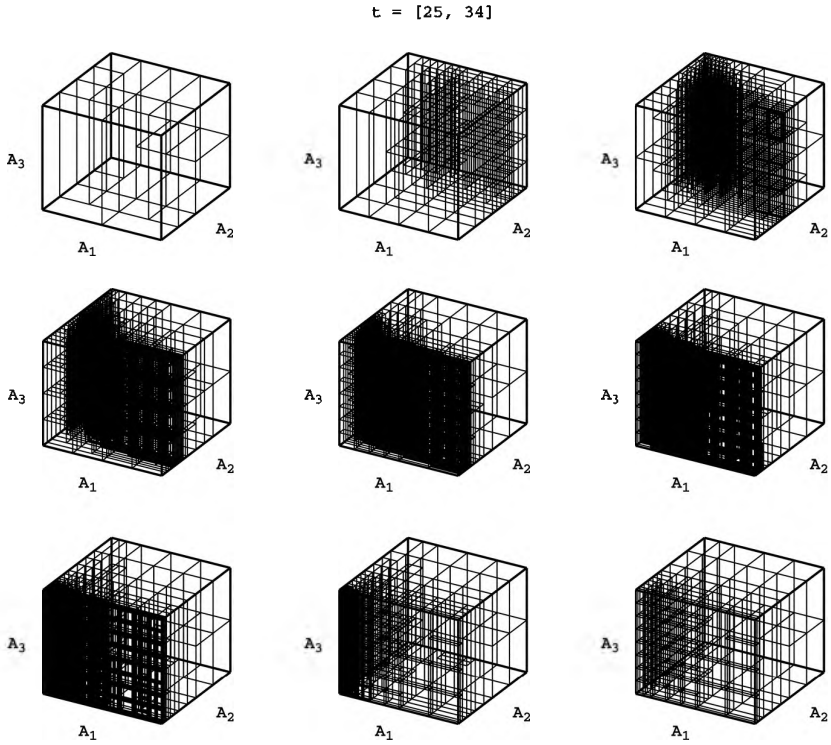

Рис. 82. Разбиения пространства в различные моменты времени

Рассмотрим расширенный механизм горения водород-кислородной смеси, включающий в себя дополнительно радикалы HO_2 и H_2O_2 и содержащий 19 реакций (табл. 4.7). В 1, 4 и 11-ю реакции внесены неопределенности. Начальные условия и параметры соответствуют предыдущему примеру. На рис. 84 представлены верхние и ниж-

ние оценки для концентраций всех компонент смеси, а на рис. 85 приведены верхние и нижние границы температуры и давления. Неопределенность в константах скоростей реакций приводит к неопределенности времени задержки воспламенения в диапазоне от 10 до 34 мкс.

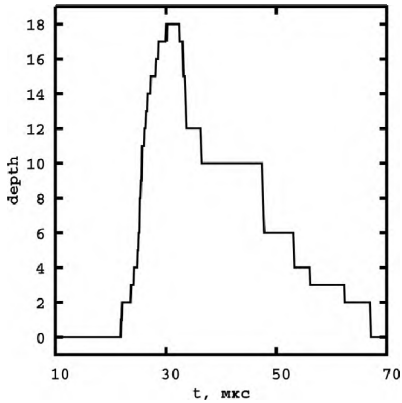

Рис. 83. Зависимость высоты kd-дерева от времени

Таблица 4.7

Расширенный механизм горения смеси H_2-O_2

Реакция	A, м, моль, с	n	E, кДж/моль
1. $O_2+H \leftrightarrow OH+O$	$[1.0\times10^8, 4.0\times10^8]$	0.0	70.3
2. $H_2+O \leftrightarrow OH+H$	5.06×10^{-2}	2.67	26.3
3. $H_2+OH \leftrightarrow H_2O+H$	1.0×10^2	1.6	13.8
4. $OH+OH \leftrightarrow H_2O+O$	$[0.75\times10^3, 3.0\times10^3]$	1.14	0.42
5. $H+H+M \leftrightarrow H_2+M$	1.8×10^6	–1.0	0.0
6. $O+O+M \leftrightarrow O_2+M$	2.9×10^5	1.0	0.0
7. $H+OH+M \leftrightarrow H_2O+M$	2.2×10^{10}	–2.0	0.0
8. $H+O_2+M \leftrightarrow HO_2+M$	2.3×10^6	–0.8	0.0
9. $HO_2+H \leftrightarrow OH+OH$	1.5×10^8	0.0	4.2
10. $HO_2+H \leftrightarrow H_2+O_2$	2.5×10^7	0.0	2.9
11. $HO_2+H \leftrightarrow H_2O+O$	$[1.5\times10^7, 6.0\times10^7]$	0.0	7.2
12. $HO_2+O \leftrightarrow OH+O_2$	1.8×10^7	0.0	–1.7
13. $HO_2+OH \leftrightarrow H_2O+O_2$	6.0×10^7	0.0	0.0
14. $HO_2+HO_2 \leftrightarrow H_2O_2+O_2$	2.5×10^5	0.0	–5.2
15. $OH+OH+M \leftrightarrow H_2O_2+M$	3.25×10^{10}	–2.0	0.0
16. $H_2O_2+H \leftrightarrow H_2+HO_2$	1.7×10^6	0.0	15.7
17. $H_2O_2+H \leftrightarrow H_2O+OH$	1.0×10^7	0.0	15.0
18. $H_2O_2+O \leftrightarrow OH+HO_2$	2.8×10^7	0.0	26.8
19. $H_2O_2+OH \leftrightarrow H_2O+HO_2$	5.4×10^6	0.0	4.20

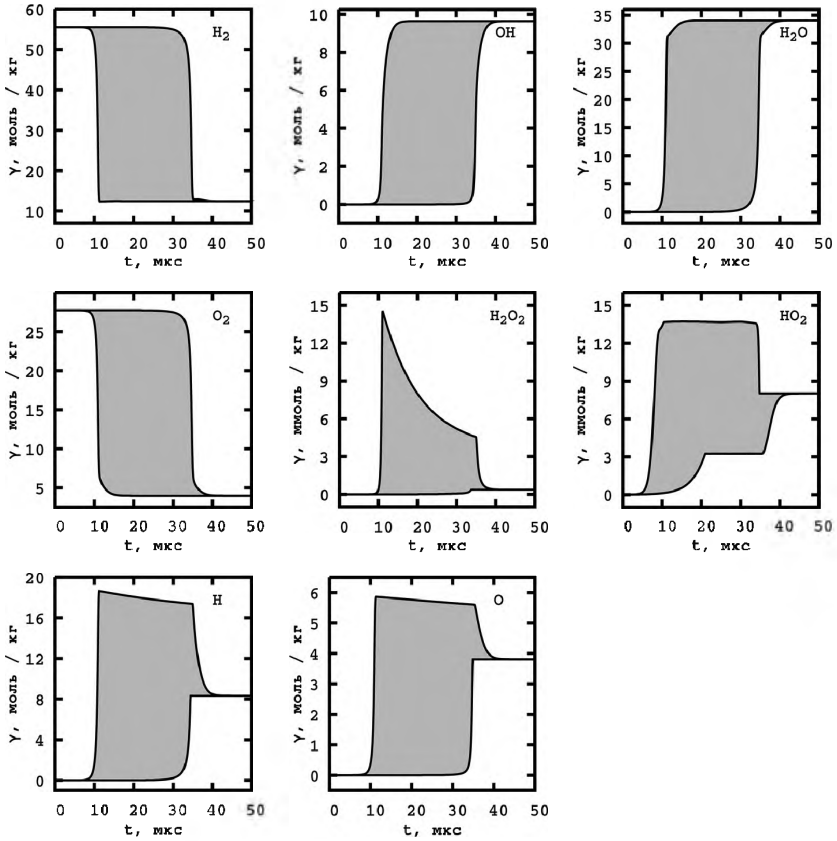

Рис. 84. Зависимость мольно-массовых концентраций H_2, OH, H_2O, O_2, H_2O_2, HO_2, H и O от времени

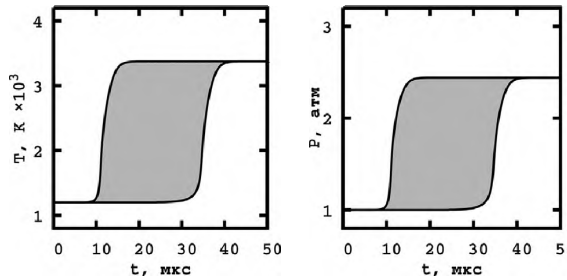

Рис. 85. Зависимость температуры и давления от времени

4.3.2. Химическое неравновесное течение в сопле

При моделировании течения в сопле жидкостного ракетного двигателя (ЖРД) неоднозначность в кинетических константах приводит к неопределенностям во всех макропараметрах, например таких, как тяга, число Маха и др. Также неоднозначными становятся параметры заморозки значений концентраций токсичных продуктов сгорания, что является важным с точки зрения экологии. Для таких задач естественной потребностью является определение интервальных оценок решений по известным интервальным значениям исходных данных. На практике моделирование течения в сопле ЖРД сводится к решению жестких систем ОДУ, которые могут быть проинтегрированы с использованием алгоритма адаптивной интерполяции.

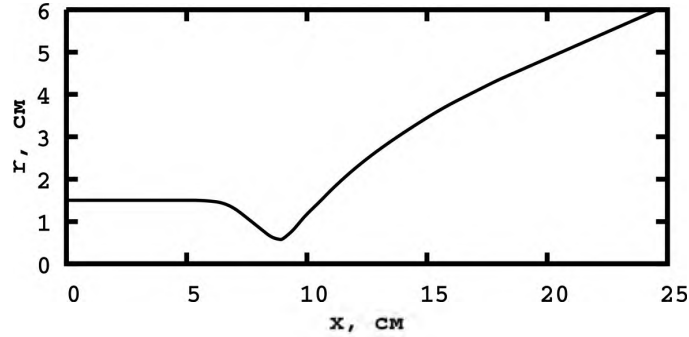

Рис. 86. Профиль сопла

Рассматривается одномерное течение в сопле жидкостного ракетного двигателя (рис. 86), работающего на несимметричном диметилгидразине $(CH_3)_2N_2H_2$ и азотном тетраксиде N_2O_4. Давление в камере сгорания $P = 100$ атм, коэффициент избытка окислителя $\alpha = 1$, энтальпия $H = 42.57$ кДж/кг. Концентрации в камере сгорания рассчитывались из условия химического равновесия. Химические процессы моделировались кинетическим механизмом [108], включающим 15 реакций, в которых участвовали 12 компонентов (табл. 4.8).

Таблица 4.8

Кинетический механизм для системы C-O-H-N

Реакция	A, м, моль, с	n	E, кДж/моль
1. $CO+O+M \leftrightarrow CO_2+M$	$[3.5\times10^2, 3.5\times10^3]$	0	1.06
2. $OH+H+M \leftrightarrow H_2O+M$	1.2×10^8	–1	0
3. $O+N+M \leftrightarrow NO+M$	3.3×10^3	0	0
4. $H+H+M \leftrightarrow H_2+M$	1.4×10^8	–1.5	0
5. $O+O+M \leftrightarrow O_2+M$	5.5×10^5	–0.87	0
6. $N+N+M \leftrightarrow N_2+M$	2.7×10^4	–0.5	0
7. $H+O+M \leftrightarrow OH+M$	3.3×10^6	–0.5	0
8. $H_2+OH \leftrightarrow H_2O+H$	1.1×10^8	0	4.33
9. $H_2+O \leftrightarrow OH+H$	1.3×10^7	0	4.96
10. $O_2+H \leftrightarrow OH+O$	2.2×10^8	0	8.3
11. $O_2+N_2 \leftrightarrow NO+NO$	5.2×10^7	0	53.85
12. $NO+N \leftrightarrow N_2+O$	3×10^7	0	0.1
13. $NO+O \leftrightarrow O_2+N$	1.1×10^7	0	20.97
14. $OH+OH \leftrightarrow H_2O+O$	1.0×10^7	0	0.6
15. $CO+OH \leftrightarrow CO_2+H$	$[2.5\times10^6, 2.5\times10^7]$	0	2.57

Рассмотрим течение газа без учета влияния вязкости, теплопроводности и диффузии. Такие течения газа в каналах с пологими стенками в областях непрерывности течения задаются уравнениями, которые имеют следующую дивергентную форму:

$$\begin{cases} \dfrac{\partial}{\partial t}\rho F + \dfrac{\partial}{\partial x}\rho u F = 0, \\ \dfrac{\partial}{\partial t}\rho u F + \dfrac{\partial}{\partial x}\left(\rho u^2 + P\right)F = P\dfrac{\partial F}{\partial x}, \\ \dfrac{\partial}{\partial t}\rho\left(e+\dfrac{u^2}{2}\right)F + \dfrac{\partial}{\partial x}\rho u\left(e+\dfrac{P}{\rho}+\dfrac{u^2}{2}\right)F = 0, \\ \dfrac{\partial}{\partial t}\rho\gamma_i F + \dfrac{\partial}{\partial x}\rho u\gamma_i F = FW_i,\ i=\overline{1,\ N}, \end{cases} \tag{4.7}$$

где

$$W_i = \sum_{r=1}^{N_r}\left(\vec{\nu}_i^{(r)} - \overleftarrow{\nu}_i^{(r)}\right)\left(\vec{W}^{(r)} - \overleftarrow{W}^{(r)}\right),$$

$$\vec{W}^{(r)} = \vec{K}^{(r)}(T)\prod_i\left(\rho\gamma_i\right)^{\vec{\nu}_i^{(r)}},\ \vec{K}(T) = AT^n\exp\left(-\frac{E}{T}\right).$$

Здесь первые три уравнения — уравнения сохранения массы (неразрывности), импульса и энергии соответственно; последние уравнения — уравнения, описывающие изменение химического состава; $F = F(x)$ — зависимость площади канала от продольной координаты; N — число компонентов в смеси. Подробно модель химической кинетики описана в предыдущем подразделе. Система уравнений замыкается термическими и калорическими уравнениями состояния: $\rho = \rho(T, P, \overline{\gamma})$, $e = e(T, P, \overline{\gamma})$.

Для моделирования течения использовался метод конечных объемов с TVD-монотонизацией, для расчетов потоков через границы ячеек — схема Хартена–Лакса–ван Лира. Так как полученная система ОДУ является жесткой, она интегрировалась в два этапа: на каждом шаге метода Рунге–Кутты второго порядка, которым интегрировались газодинамические уравнения, выполнялось несколько шагов неявным методом Розенброка с заморозкой матрицы Якоби для интегрирования уравнений химической кинетики в каждой ячейке.

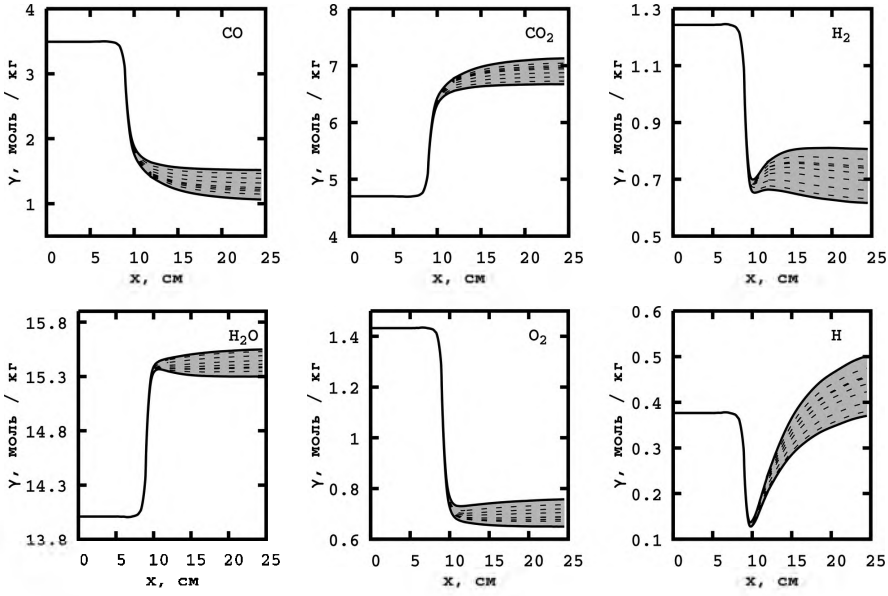

Рис. 87. Распределение мольно-массовых концентраций CO, CO_2, H_2, H_2O, O_2 и H в сопле

На рис. 87 отражены распределения концентрации некоторых компонентов смеси в сопле после установления течения. Неопределенности в константах скоростей реакций сказываются на заморозке компонентов смеси, что, в свою очередь, отражается на экологичности двигателя. На рисунке пунктирными линиями показано некоторое количество выполненных методом Монте-Карло запусков. Все они содержатся в полученных оценках. На рис. 88 показаны интервальные оценки макропараметров. В отличие от концентраций, неопределенности в константах скоростей в значительно меньшей мере влияют на число Маха, температуру, давление и т.д.

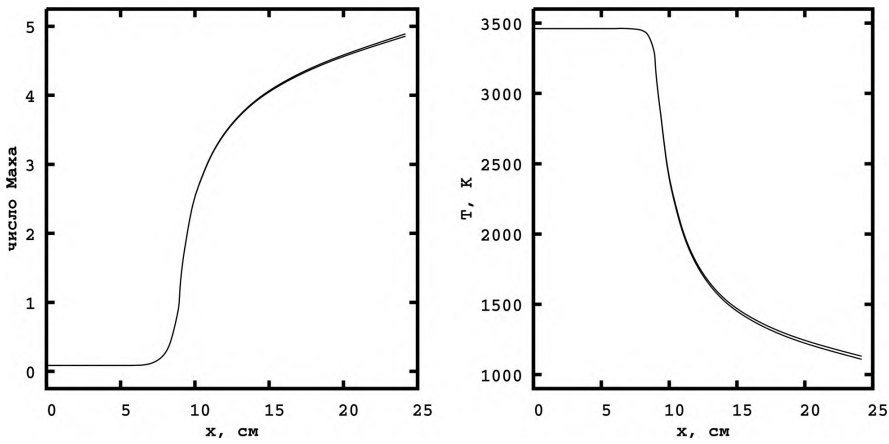

Рис. 88. Распределение макропараметров в сопле

4.3.3. Стоячая детонационная волна

В соответствии с классической теорией, детонационная волна (ДВ), распространяющаяся по взрывчатой смеси, представляет собой совокупность ударной волны (УВ) и примыкающей к ней тонкой зоны, в которой протекают экзотермические химические реакции, заканчивающиеся установлением химического равновесия, причем зона химических превращений распространяется со скоростью УВ.

В этом подразделе рассматривается одномерное сверхзвуковое течение в сопле, для которого характерно возникновение стоячей

детонационной волны. Интерес представляет случай, когда стоячая ДВ реализуется до критического сечения (КС) и после этого поток вновь ускоряется до сверхзвуковой скорости при переходе через КС. Согласно классической теории [109], стоячая ДВ в сужающейся части канала является неустойчивой, поэтому для ее длительного стабильного существования используется форма сопла с двумя сужениями. Математическая модель представляет собой систему уравнений (4.7).

Сначала рассмотрим форму сопла, в которой первое сужение слабо выражено (рис. 89). Диапазон, в котором находится радиус на участке расширения, подобран исходя из решения соответствующей стационарной равновесной задачи. В этом диапазоне гарантированно находится стоячая ДВ, если предположить, что все химические превращения происходят бесконечно быстро. В предыдущей задаче было получено, что неопределенности в константах скоростей реакций влияют на макропараметры незначительно, и для того, чтобы усилить от них эффект, участок сопла, где должна установиться ДВ, сделан очень пологим.

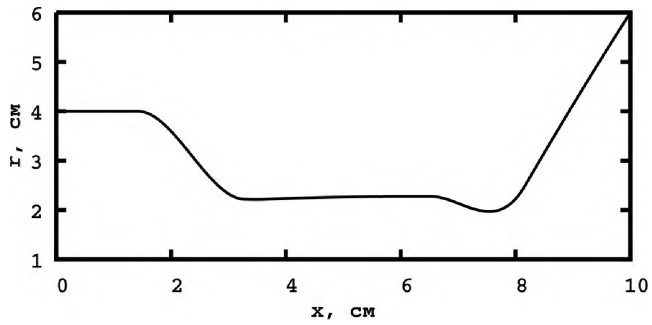

Рис. 89. Профиль сопла с двумя сужениями

На вход в сопло со скоростью $u = 2750$ м/с, давлением $P = 1$ атм и температурой $T = 400$ K подается смесь H_2, O_2, N_2, Ar в соотношении 42 : 21 : 78 : 1 [110]. В кинетический механизм внесены четыре интервальные неопределенности (табл. 4.9), которые замедляют соответствующие реакции в сто тысяч раз.

Таблица 4.9

Кинетический механизм для системы H_2-O_2

Реакция	A, м, моль, с	n	E, К
1. $H_2O+H \leftrightarrow OH+H_2$	$[8.4\times10^2, 8.4\times10^7]$	0	10116
2. $O_2+H \leftrightarrow OH+O$	$[2.2\times10^3, 2.2\times10^8]$	0	8455
3. $H_2+O \leftrightarrow OH+H$	1.8×10^4	1	4480
4. $O_2+M \leftrightarrow 2O+M$	5.4×10^{12}	–1	59400
5. $H_2+M \leftrightarrow 2H+M$	2.2×10^8	0	48300
6. $H_2O+M \leftrightarrow OH+H+M$	$[1\times10^{13}, 1\times10^{18}]$	–2.2	59000
7. $HO+M \leftrightarrow O+H+M$	8.5×10^{12}	–1	50830
8. $H_2O+O \leftrightarrow 2OH$	$[5.8\times10^2, 5.8\times10^7]$	0	9059
9. $H+O_2+M \leftrightarrow HO_2+M$	3.5×10^4	–0.41	–565
10. $H_2+O_2 \leftrightarrow H+HO_2$	7.39×10^{-1}	0.6	26926
11. $H_2O+O \leftrightarrow H+HO_2$	4.76×10^5	2.43	28743
12. $H_2O+O_2 \leftrightarrow OH+HO_2$	1.5×10^9	0.372	36600
13. $2OH \leftrightarrow H+HO_2$	1.2×10^7	0.5	20200
14. $OH+O_2 \leftrightarrow O+HO_2$	1.3×10^7	0	28200
15. $H+H_2O_2 \leftrightarrow H_2+HO_2$	1.6×10^6	0	1900
16. $H+H_2O_2 \leftrightarrow H_2O+OH$	5×10^8	0	5000
17. $2HO_2 \leftrightarrow H_2O_2+O_2$	1.8×10^7	0	500
18. $HO_2+H_2O \leftrightarrow H_2O_2+OH$	1.8×10^7	0	15100
19. $OH+HO_2 \leftrightarrow H_2O_2+O$	5.2×10^4	0.5	10600
20. $H_2O_2+M \leftrightarrow 2OH+M$	1.2×10^{11}	0	22900

Установление течения выполнялось в несколько этапов. На первом этапе было получено только сверхзвуковое течение нереагирующего газа. На втором этапе искусственным образом провоцировалась УВ. В момент, когда она оказалась на нужном участке сопла, подключался неинтервальный кинетический механизм. На третьем этапе, уже после установления стоячей ДВ, был использован интервальный кинетический механизм.

На рис. 90 изображены распределения концентраций компонентов в сопле. Присутствие неопределенностей в кинетическом механизме сильно повлияло на ширину интервальных оценок концентраций, в отличие от ширины оценок макропараметров (рис. 91, 92). Тем не менее здесь наблюдается небольшой сдвиг фронта стоячей ДВ (рис. 91). Присутствующие пиковые всплески концентраций на последних

трех графиках не являются аномальными и объясняются тем, что в основном все химические превращения происходят в области ДВ.

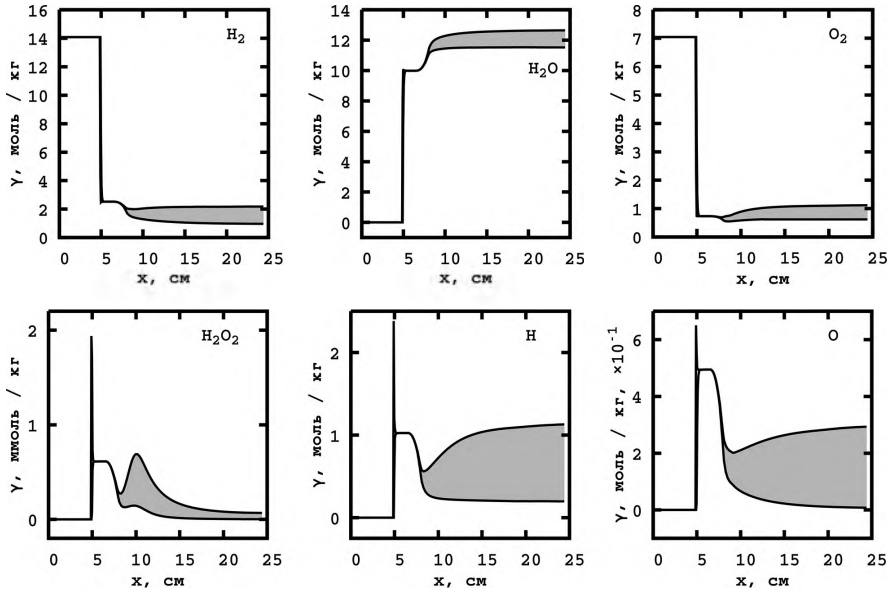

Рис. 90. Распределение мольно-массовых концентраций H_2, H_2O, O_2, H_2O_2, H и O в сопле

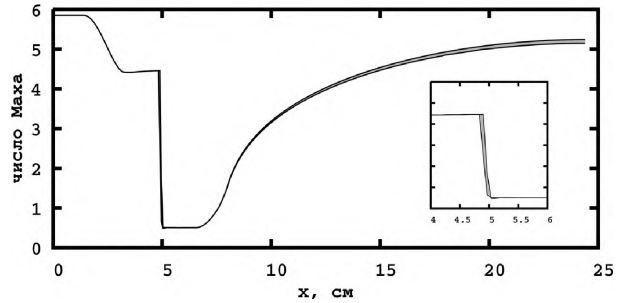

Рис. 91. Распределение числа Маха в сопле

На следующем примере более детально рассмотрим структуру стоячей ДВ и то, как на нее влияют небольшие неопределенности в константах скоростей реакций. Как и раньше, здесь используется

сопло с двумя сужениями (рис. 93) и на вход подается такая же смесь из H_2, O_2, N_2, Ar. В том месте, где примерно должна установиться ДВ, выполняется сильное уплотнение расчетной сетки, чтобы можно было подробно рассмотреть ДВ. С учетом построенной пространственной сетки полученная система ОДУ содержит в себе более 30 000 уравнений. Скорость на входе в сопло $u = 2500$ м/с, давление $P = 1.38$ атм и температура $T = 391$ K. В табл. 4.9 вместо четырех интервальных коэффициентов присутствуют только два из табл. 4.10. На рис. 94 представлены распределения интервальных оценок концентраций всех веществ в сопле в окрестности стоячей ДВ.

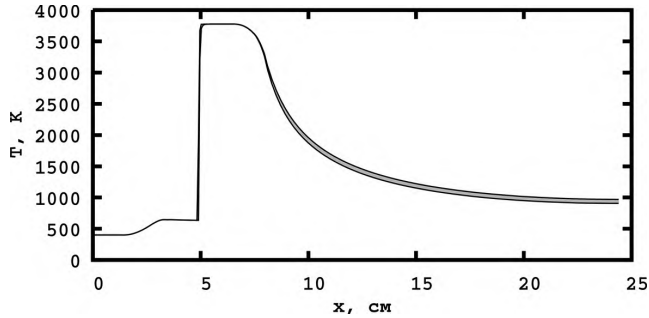

Рис. 92. Распределение температуры в сопле

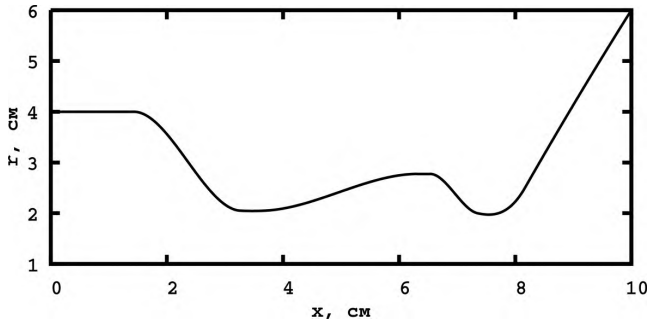

Рис. 93. Профиль сопла

Полученные решения имеют схожесть с графиками на рис. 84, где моделировался процесс горения смеси водорода и кислорода с использованием другого кинетического механизма.

Таблица 4.10

Интервальная часть механизма из табл. 4.9 и 4.7

Реакция	A, м, моль, с	n	E, К
16. $H+H_2O_2 \leftrightarrow H_2O+OH$	$[2.5\times10^8, 5\times10^8]$	0	5000
19. $OH+HO_2 \leftrightarrow H_2O_2+O$	$[5.2\times10^4, 2.6\times10^5]$	0.5	10600

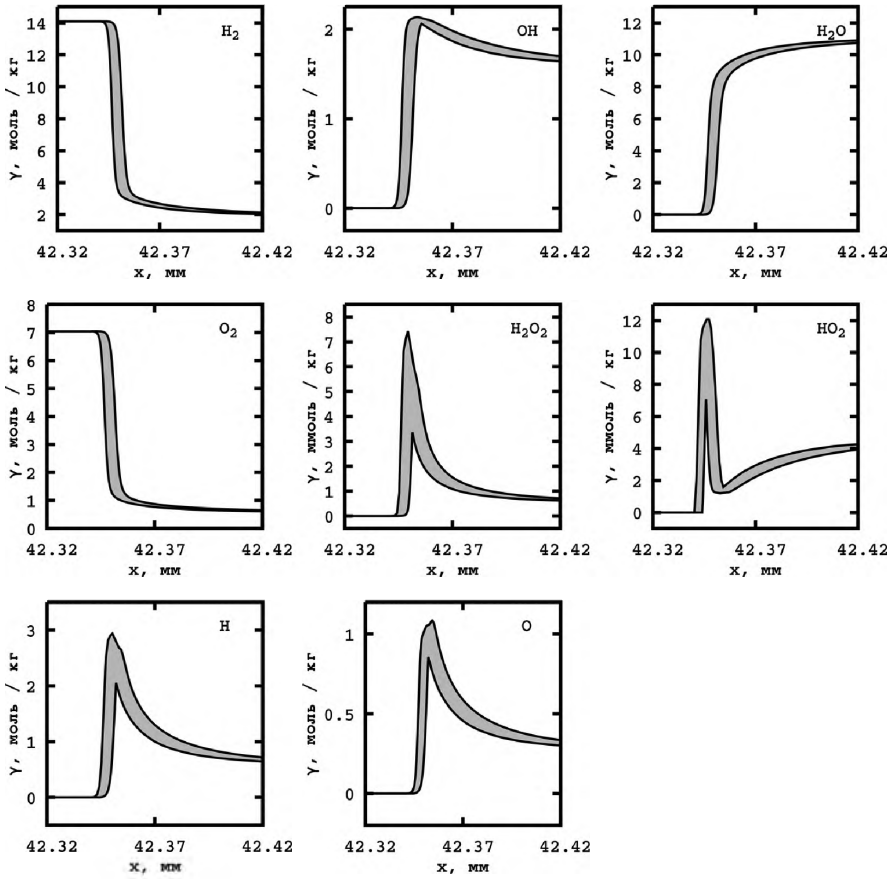

Рис. 94. Распределение концентраций компонентов в области стоячей ДВ

На рис. 95 и 96 на левых графиках изображено распределение температуры и числа Маха в сопле. Практически незаметно, что и

температура, и число Маха являются интервальными. При увеличении масштаба по оси x в 2500 раз (правые графики) становится видна структура ДВ. На графике температуры четко различима УВ и последующая за ней зона экзотермических химических реакций. Здесь присутствует небольшое смещение фронта ДВ и небольшое увеличение времени задержки воспламенения.

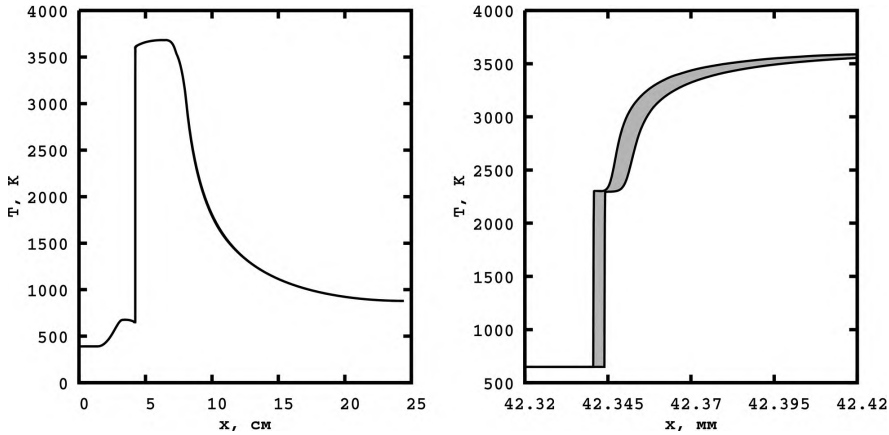

Рис. 95. Распределение температуры в области стоячей ДВ

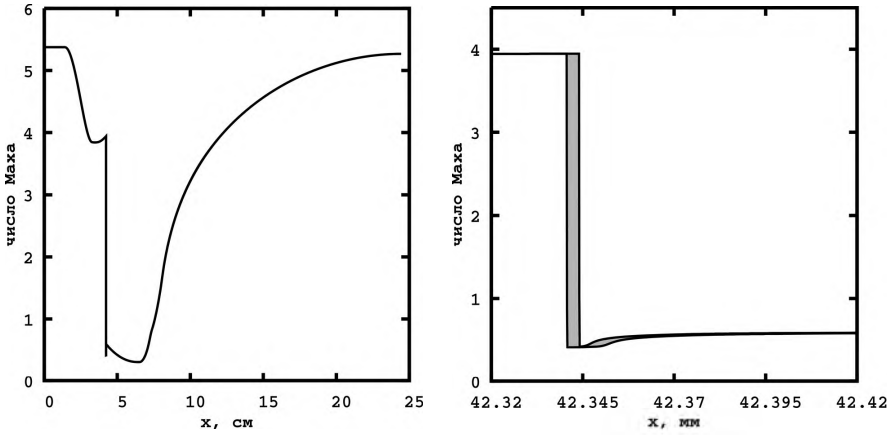

Рис. 96. Распределение числа Маха в области стоячей ДВ

4.4. ЗАКЛЮЧЕНИЕ

Алгоритм адаптивной интерполяции апробирован на различных прикладных и исследовательских задачах:

- Динамические системы, в которых имеют место бифуркации и хаос. По качественному изменению структуры адаптивной сетки, получающейся в процессе вычислений, можно определять, какие режимы возникают в динамической системе. За счет тонкой настройки алгоритма для систем с хаосом появляется возможность проводить исследования в течение продолжительного времени.
- Задачи небесной механики. Выполнен расчет трубки траекторий для астероида FX11 в условиях неопределенности положения астероида и его скорости. Полученные результаты демонстрируют превосходство алгоритма адаптивной интерполяции с точки зрения точности и вычислительных затрат по сравнению с различными программными комплексами.
- Задачи химической кинетики и газовой динамики. Проведено моделирование горения водород-кислородной смеси при наличии неопределенностей в константах скоростей реакций. Разработана математическая модель неравновесных течений с учётом неопределенности значений констант скоростей реакций. Выполнены численные исследования влияния неопределенностей на структуру детонационной волны, а так же на параметры установившегося течения, такие как время задержки воспламенения и концентрация вредных веществ на выходе из сопла.

Все полученные результаты не противоречат уже известным решениям и совпадают с решениями, полученными существенно более затратным методом Монте-Карло.

ГЛАВА 5. ПЕРСПЕКТИВЫ. БОЛЬШИЕ РАЗМЕРНОСТИ

Описанный алгоритм адаптивной интерполяции при всех своих достоинствах (универсальность, робастность, точность и возможность распараллеливания) обладает одним существенным недостатком: с увеличением количества интервальных параметров в задаче его сложность возрастает экспоненциально. В каждой вершине kd-дерева содержится интерполяционная сетка с количеством узлом $(p+1)^m$, где p — степень интерполяционного полинома по каждому измерению, а m — количество интервальных начальных условий. Уже при параметрах $p = 4$ и $m = 20$ количество узлов в сетке будет порядка ста триллионов ($\approx 10^{14}$). Если рассматривать полные кинетические механизмы, в которых реакции исчисляются тысячами, то приходится иметь дело с тысячемерными областями неопределенности параметров, и использование алгоритма адаптивной интерполяции в этом случаи становится невозможным.

Улучшить данную ситуацию можно, сделав предположение, которое вполне соответствует реальности: на практике редко бывает ситуация, когда абсолютно все параметры в отдельности или в совокупности оказывают существенное влияние на решение. При построении интерполяционного полинома, например, для функции двух переменных

$$P(x,y) = a_{0,0} + a_{1,0}x + a_{0,1}y + a_{1,1}xy + \ldots = \sum_{i=0}^{p}\sum_{j=0}^{p} a_{i,j}x^i y^j \ .$$

Это означает, что большая часть членов $a_{i,j}x^i y^j$ не будут вносить существенный вклад в результат и, следовательно, их можно не учитывать. Поэтому для построения интерполяционного полинома достаточно использовать не все $(p+1)^2$ узлов.

Возникает несколько вопросов. Какие именно члены необходимо учитывать? Как выбираются узлы для интерполяционной сетки? Априори ответить однозначно на эти вопросы нельзя.

5.1. КРЕСТОВАЯ АППРОКСИМАЦИЯ

Рассмотрим интерполяционный полином Лагранжа на регулярной сетке для двух переменных. Вычисление значения в некоторой точке

сводится к поэлементному перемножению двух матриц и суммированию всех элементов:

$$P(x,y) = L(x,y) \otimes F = \sum_{i=0}^{p} \sum_{j=0}^{p} l_{i,j}(x,y) f_{i,j} \ ,$$

где $l_{i,j}(x,y)$ — базисные полиномы Лагранжа, $f_{i,j} = f(x_i, y_j)$ — значения искомой функции в узлах сетки x_i, y_j. Основная цель — уменьшение количества вычислений функции f.

Сделанное в начале главы предположение, по сути, означает, что ранг матрицы F может быть меньше, чем $(p + 1)$. Допустим, что функция f имеет следующую структуру:

$$f(x,y) = u_1(x)v_1(y) + u_2(x)v_2(y) + \ldots + u_r(x)v_r(y) \, ,$$

тогда матрица значений $F = \{f_{i,j}\}$ может быть представлена в виде разложения:

$$F = \sum_{i=1}^{r} \begin{pmatrix} u_i(x_0) \\ u_i(x_1) \\ \ldots \\ u_i(x_p) \end{pmatrix} \begin{pmatrix} v_i(y_0) & v_i(y_1) & \ldots & v_i(y_p) \end{pmatrix} = UV \, .$$

Важный факт: если матрица имеет ранг r, то, зная r строк и r столбцов, можно полностью восстановить всю матрицу. Этот факт лежит в основе рассматриваемого подхода в первых трех разделах.

Здесь тоже возникает ряд вопросов. Какие именно строки и столбцы брать? Каким образом определять ранг r? Конечно, без какой-либо информации о функции F, без вычисления всех элементов матрицы гарантированно ответить на эти вопросы нельзя, поэтому отчасти все алгоритмы носят эвристический характер. На практике выполняется переход от точного разложения к аппроксимации с некоторой точностью ε:

$$\|F - UV\| < \varepsilon \, .$$

Известен ряд работ, посвященных методам построения таких разложений [111–113]. В литературе встречаются следующие названия: скелетное разложение, крестовая аппроксимация, малоранговая аппроксимация.

В простейшем виде псевдокод алгоритма представлен ниже. На вход подается матрица F размером n на m, а также требуемая абсолютная точность аппроксимации *eps*. На выходе две матрицы U и V размерами n на r и r на m соответственно, где r — ранг матрицы, который определяется в процессе вычислений. Алгоритм начинается с выбора произвольного столбца (например, с номером $m/2$). Далее попеременно выполняется «обнуление» то некоторого столбца, то некоторой строки до тех пор, пока модуль максимального элемента не станет меньше *eps*.

```
r = 0
jr = m / 2
ii = {1, 2, ..., n}
jj = {1, 2, ..., m}
jj = jj / jr
while r < min(n, m):
   ir = -1
   for i = 1, ..., n:
      U[i][r] = F[i][jr];
      for k = 1, ..., r:
           U[i][r] -= U[i][k] * V[k][jr]
      if (i in ii) and (ir < 0 or |U[i][r]| > |U[ir][r]|):
           ir = i
   if ir < 0 or |U[ir][r]| < eps:
      break
   ii = ii / ir
   jr = -1
   for j = 1, ..., m:
      V[r][j] = F[ir][j]
      for k = 1, ..., r:
           V[r][j] -= U[ir][k] * V[k][j]
      V[r][j] /= U[ir][r]
      if (j in jj) and (jr < 0 or |V[r][j]| > |V[r][jr]|):
           jr = j
   if jr < 0 or |V[r][jr]| < eps:
      r += 1
      break
   jj = jj / jr
   r += 1
return U, V, r
```

Рассмотрим систему ОДУ:

$$\begin{cases} x' = y, \\ y' = -\sin(x), \\ x(0) = x_0 \in [-1.0, 1.0], \\ y(0) = y_0 \in [0.0, 1.0]. \end{cases}$$

Над множеством образованным интервальными начальными условиями вводиться регулярная сетка: $x_0^i = -1 + 0.02i$, $y_0^j = 0.01j$, $0 \le i \le 100$, $0 \le j \le 100$. Пусть матрица $\underset{101\times101}{F}$ состоит из значений фазовой переменной x в момент времени $t = 30$:

$$f_{i,j} = x_{i,j}(30) \mid x_0 = -1 + 0.02i,\ y_0 = 0.01j\ .$$

На рис. 97 линиями показаны те строки и те столбцы, которые понадобились в процессе построения разложения. Если вся матрица состоит из 10 201 элемента, то здесь потребовалось вычислить всего 2101 элемент (11 строк и столбцов: $2\times11\times101 - 11\times11 = 2101$), то есть пятую часть от всей матрицы, чтобы построить аппроксимацию с $eps = 10^{-5}$.

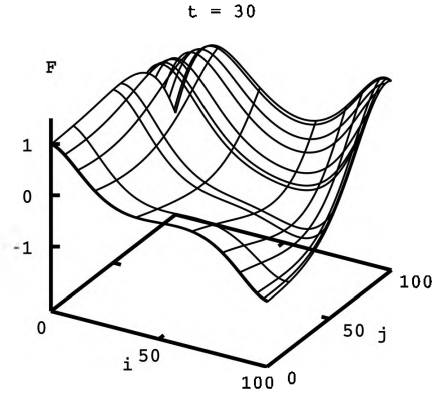

Рис. 97. Строки и столбцы, по которым восстанавливается вся матрица

Если предположить, что все элементы матрицы известны, то вопрос о существовании разложения и его нахождении является решенным: однозначным образом строится SVD-разложение, из которого отбрасываются строки и столбцы, соответствующие несущественным значениям сингулярных чисел.

5.2. ТЕНЗОРНЫЙ ПОЕЗД

Для матриц, то есть для случая двух переменных, вопрос является достаточно изученным и разработанным: существуют эффективные методы и алгоритмы для построения разложений. В случае большего количества измерений необходимо работать с многомерными массивами, с тензорами. Общая идея эффективного представления этих объектов основывается на разделении переменных. Существует несколько подходов: каноническое разложение [114], разложение Такера [115] и TT-разложение [116, 117] (tensor train, тензорный поезд). Для канонического разложения не существует надежных алгоритмов, а разложение Такера сложно применимо для большого числа измерений. TT-разложение появилось сравнительно недавно, и отличительной его особенностью является неподверженность проклятию размерности.

Пусть дан n-мерный тензор

$$A \in \mathbb{R}^{p_1 \times p_2 \times \ldots \times p_n}.$$

Его TT-разложение записывается следующим образом:

$$\hat{A}(i_1, i_2, \ldots, i_n) = G_1(i_1) G_2(i_2) \ldots G_n(i_n) \ ,$$

где

$$G_k \in \mathbb{R}^{p_k \times r_{k-1} \times r_k}, 1 \le i_k \le p_k, 1 \le k \le n, r_0 = r_n = 1 .$$

Это есть произведение n трехмерных тензоров (рис. 98). Для вычисления значения конкретного элемента выполняется перемножение соответствующих матриц.

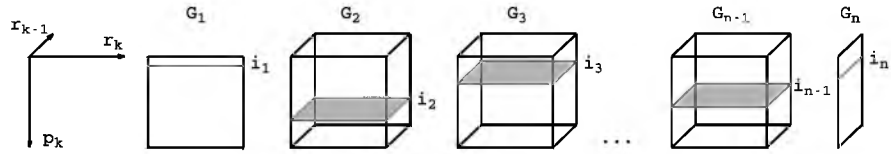

Рис. 98. Иллюстрация TT-разложения

Построение TT-разложения сводится к обычным матричным разложениям. Сначала выполняется переход от n измерений к двум, с помощью простого преобразования индексов: первый индекс остается как есть, а остальные индексы образуют второй матричный индекс.

Для матрицы вычисляется SVD-разложение, и полученные матрицы превращаются обратно в тензоры меньшей размерности, для которых рекурсивно можно применить алгоритм.

Однако SVD-разложение требует знания всех элементов в матрице, что затрудняет его применение для решения поставленной задачи. Кроме того, оно является дорогим с точки зрения вычислительных затрат. В [116] разработано обобщение крестовой аппроксимации на многомерный случай: алгоритм TT-cross. Основная идея заключается в замене SVD-разложения на малоранговую аппроксимацию. Алгоритм, использующийся для этого, отличается от алгоритма, приведенного ранее в главе. В его основе лежит нахождение подматрицы максимального объема. При этом ранги получающихся матриц подбираются постепенно: выполняется построение TT-разложения; оценивается погрешность, и если она оказывается неприемлемой, то ранги увеличиваются и алгоритм запускается сначала.

Помимо неподверженности «проклятию размерности», важным свойством TT-разложения является то, что большинство арифметических операций, и не только арифметических, можно выполнять над тензорами, находясь в этом формате. Это, например, нахождение суммы всех элементов, поиск максимума/минимума, поэлементное сложение/вычитание/умножение/деление двух тензоров и т.п. Существует несколько реализаций TT-разложения. Основной библиотекой считается ttpy, разработанная авторами этого подхода на языке программирования python. В ней есть все необходимые операции и методы для работы с тензорами в TT-формате.

Благодаря данному представлению тензоров появилась возможность работать на обычных пользовательских компьютерах с объектами, содержащими элементов больше, чем атомов в Солнечной системе. Конечно, важным моментом является то, что исходные данные обладают колоссальной избыточностью, которая устраняется этим методом.

Рассмотрим задачу, описанную в предыдущей главе. Проведем более полное сопоставление кинетических механизмов, приведенных в работах [105, 106] и представленных в табл. 4.7 и 4.9. На рис. 99 показаны зависимости констант скоростей от температуры для некоторых реакций. Серым цветом проиллюстрирован разброс величин по данным разных источников.

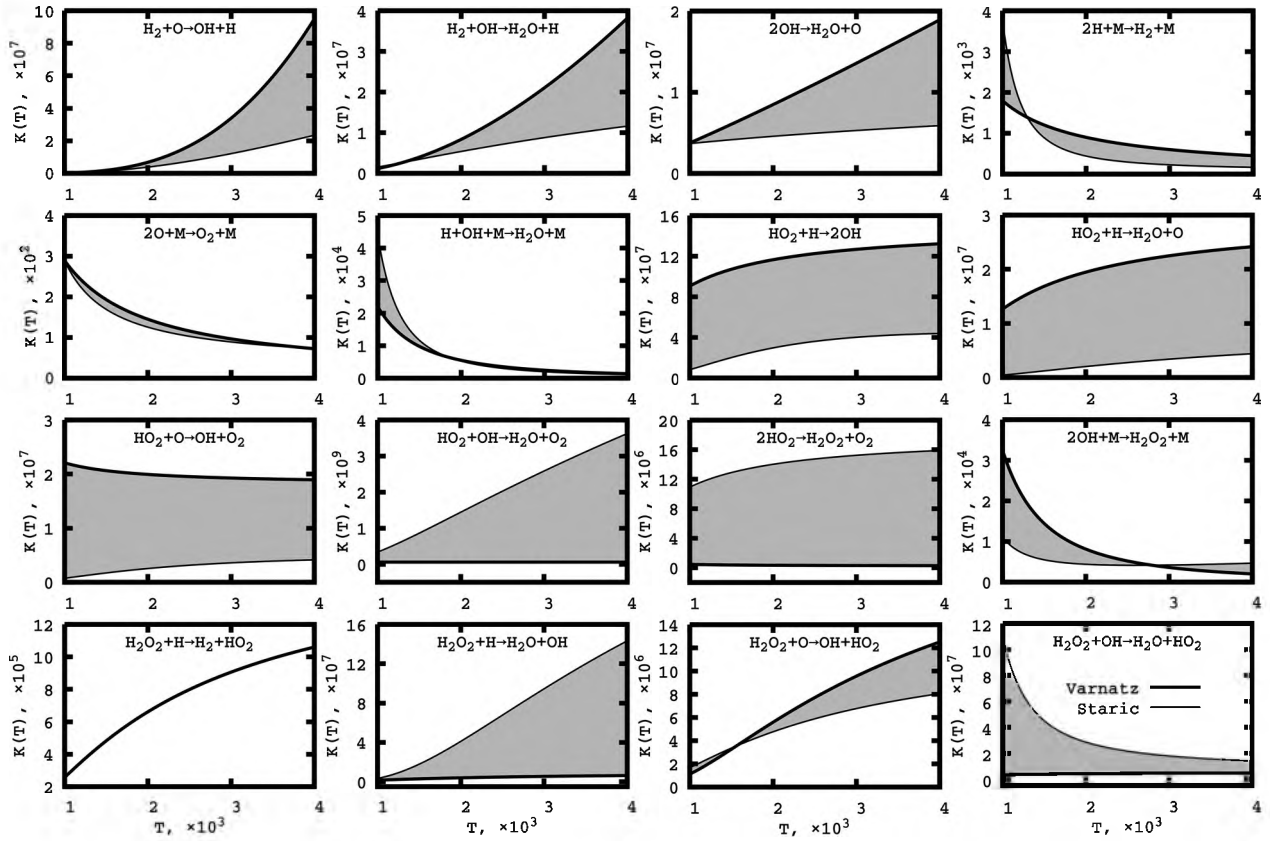

Рис. 99. Сравнение расширенных механизмов горения смеси H_2-O_2

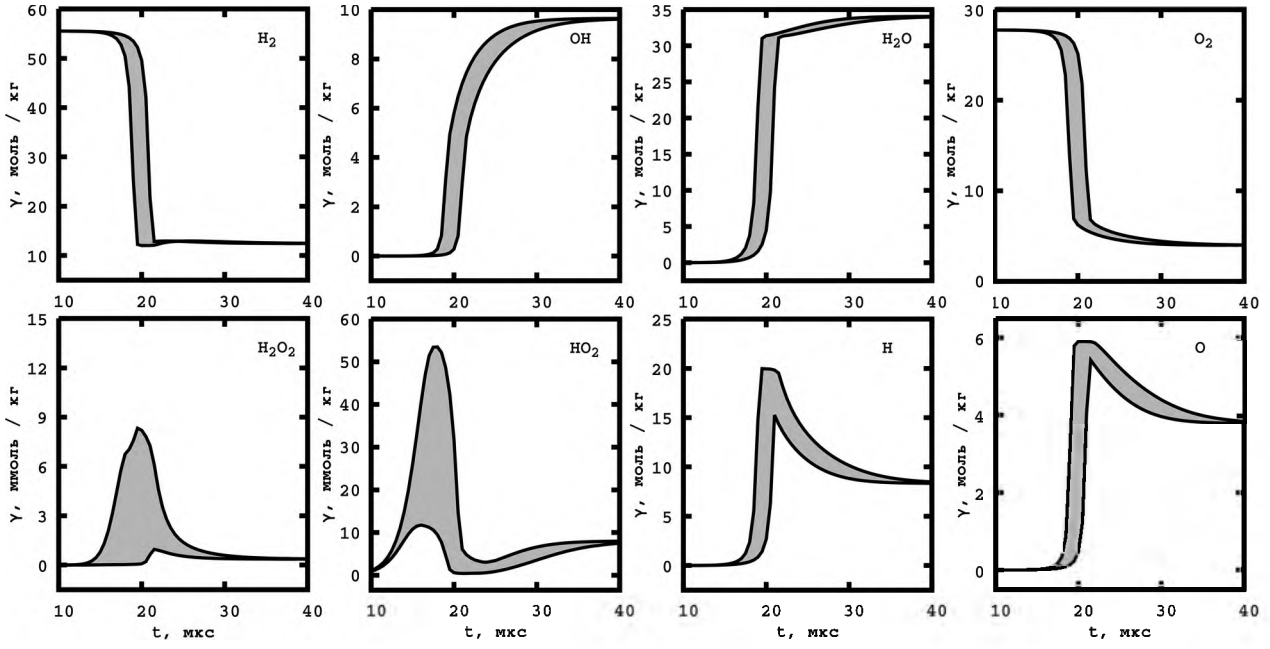

Рис. 100. Зависимость оценок концентраций компонент смеси от времени

В предыдущей главе в качестве интервального параметра выступал предэкспоненциальный множитель в формуле Аррениуса. Здесь же используется более точная аппроксимация: в качестве интервального коэффициента выступает коэффициент суперпозиции значений констант скоростей реакций из разных механизмов:

$$K(T) = (1 - a)K_1(T) + aK_2(T),\ a \in [0, 1].$$

Таким образом, получается шестнадцатимерный куб $[0, 1]^{16}$, над которым строится регулярная сетка. По каждому измерению делается 32 шага дискретизации, в сумме в тензоре содержится $32^{16} \approx 10^{24}$ элементов. Моделируется горение водородно-воздушной смеси с параметрами, приведенными в разделе 4.3.1. На рис. 100 показаны зависимости верхних и нижних оценок концентраций от времени. На графике концентрации H_2O_2 присутствует некоторая угловатость, это связано с тем, что в каждый момент времени максимум и минимум брался из значений, содержащихся в тензоре без дополнительной уточняющей интерполяции.

В результате благодаря использованию алгоритма TT-cross за приемлемое вычислительное время построено TT-разложение тензора, по которому получены интервальные оценки концентраций.

5.3. АЛГОРИТМ АДАПТИВНОЙ ИНТЕРПОЛЯЦИИ И TT-РАЗЛОЖЕНИЕ

В каждой вершине дерева хранится многомерный массив (тензор). Количество элементов в тензоре зависит от количества интервальных параметров как $(p + 1)^m$. Основная идея практического улучшения этой ситуации заключается в нахождении TT-разложения, которое можно построить с помощью алгоритма TT-cross, не вычисляя всех элементов тензора.

В процессе работы алгоритма над вершинами производятся следующие операции: интерполяция по сетке и разбиение вершины на две. В первом случае для каждой точки, в которой необходимо посчитать приближенное значение, строится тензор, состоящий из значений базисных полиномов Лагранжа, в TT-формате. Далее полученный тензор поэлементно умножается на тензор, хранящийся в вершине, и в конце вычисляется сумма всех элементов. Во втором случае при

разбиении вершины требуется применить одномерную интерполяцию по конкретному измерению, чтобы построить недостающие значения в узлах новых вершин. Данное действие тоже может быть представлено в виде композиций нескольких операций, доступных в TT-формате.

В оригинальной версии алгоритма порядок p и, соответственно, размер интерполяционной сетки определялся из конкретных соображений относительно устойчивости и вычислительной сложности. Здесь же в некоторых случаях целесообразно создавать сетку, в которой узлов будет больше, чем требуется для заданного порядка p. Разбиение вершины на две может оказаться более затратной операцией с вычислительной точки зрения, чем увеличение количества узлов в рамках одной сетки.

В качестве примера рассмотрим модельную задачу: движение тел с неопределенностями в начальных скоростях под действием сил гравитации. Система ОДУ в безразмерных переменных имеет следующий вид:

$$
\begin{cases}
\left(v_i^x\right)' = \sum\limits_{\substack{j=1 \\ j\neq i}}^{7} m_j \dfrac{x_j - x_i}{\left((x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2\right)^{3/2}}, \\
\left(v_i^y\right)' = \sum\limits_{\substack{j=1 \\ j\neq i}}^{7} m_j \dfrac{y_j - y_i}{\left((x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2\right)^{3/2}}, \\
\left(v_i^z\right)' = \sum\limits_{\substack{j=1 \\ j\neq i}}^{7} m_j \dfrac{z_j - z_i}{\left((x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2\right)^{3/2}}, \\
x_i' = v_i^x,\ y_i' = v_i^y,\ z_i' = v_i^z,\ i = 1,\ldots,7, \\
x_1(0) = y_1(0) = z_1(0) = v_1^x(0) = v_1^y(0) = v_1^z(0) = 0,\ m_1 = 10^5, \\
x_2(0) = 1,\ y_2(0) = z_2(0) = 0,\ v_2(0) = \begin{pmatrix} 0 & v & 0 \end{pmatrix}^T + dv_2^T, \\
x_3(0) = -1,\ y_3(0) = z_3(0) = 0,\ v_3(0) = \begin{pmatrix} 0 & -v & 0 \end{pmatrix}^T + dv_3^T, \\
y_4(0) = 1,\ x_4(0) = z_4(0) = 0,\ v_4(0) = \begin{pmatrix} 0 & 0 & v \end{pmatrix}^T + dv_4^T, \\
y_5(0) = -1,\ x_5(0) = z_5(0) = 0,\ v_5(0) = \begin{pmatrix} 0 & 0 & -v \end{pmatrix}^T + dv_5^T, \\
z_6(0) = 1,\ x_6(0) = y_6(0) = 0,\ v_6(0) = \begin{pmatrix} v & 0 & 0 \end{pmatrix}^T + dv_6^T, \\
z_7(0) = -1,\ x_7(0) = y_7(0) = 0,\ v_7(0) = \begin{pmatrix} -v & 0 & 0 \end{pmatrix}^T + dv_7^T, \\
v = 316.23,\ m_i = 10^{-5}, dv_i \in [-1.0, 1.0]^3,\ i = 2,\ldots,7.
\end{cases}
$$

На рис. 101 прямоугольными параллелепипедами проиллюстрированы области неопределенности в пространстве для каждого тела в различные моменты времени. Данная система является показательной, потому что неопределенность в скорости конкретного тела влияет в основном только на положение и скорость этого же тела, а на другие тела влияет слабо. Параметр $p = 4$, количество элементов в тензоре в каждой вершине равно $5^{18} \cdot 42 \approx 10^{15}$. Число 42 соответствует количеству фазовых переменных.

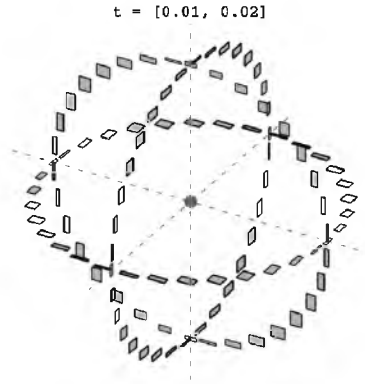

Рис. 101. Неопределенности в положении тел в различные моменты времени

За счет наличия избыточности в данных, для построения TT-разложения требуется только малая часть элементов исходного тензора. В целом, за счет использования алгоритма TT-cross получается очень эффективно уменьшить вычислительные затраты и применять алгоритм адаптивной интерполяции для задач с большим количеством интервальных параметров.

5.4. РАЗРЕЖЕННЫЕ СЕТКИ

При рассмотрении подходов к уменьшению «проклятия размерности» нельзя не упомянуть разреженные сетки Смоляка [118–122] (sparse grids). Они появились еще в 1960-х годах, для решения многопараметрических задач в экономике. При интерполяции используется кусочно-линейный иерархический базис (рис. 102).

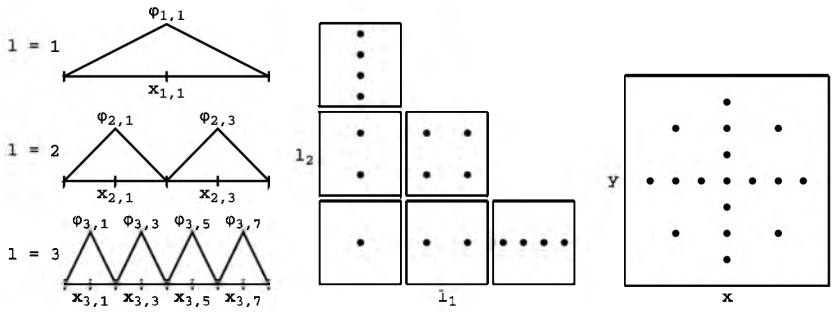

Рис. 102. Разреженные сетки. Иерархический базис

Существует адаптивный вариант данных сеток. Для структуризации используется бинарное дерево. В классическом варианте если исследуемая функция имеет отличное от нуля значение на границе области, то выполняется рассмотрение всех граней меньшей размерности. Для каждой грани строится своя адаптивная сетка. Несложно посчитать, что для n-мерной области число граней меньшей размерности будет равняться $3^n - 1$. С учетом дублирования узлов в разных сетках, в лучшем случае количество узлов будет равно 3^n, что говорит об экспоненциальной сложности данного подхода. Важным свойством разреженных сеток является гибкость адаптации: для небольшого увеличения точности в каждом конкретном случае нет необходимости увеличивать сразу в два раза количество узлов, как это было бы в алгоритме адаптивной интерполяции.

Рассмотрим примеры. На рис. 103 показаны несколько функций $\mathbb{R}^2 \to \mathbb{R}$ и получающаяся адаптивная сетка, на рис. 104 приведены сетки для функций $\mathbb{R}^3 \to \mathbb{R}$.

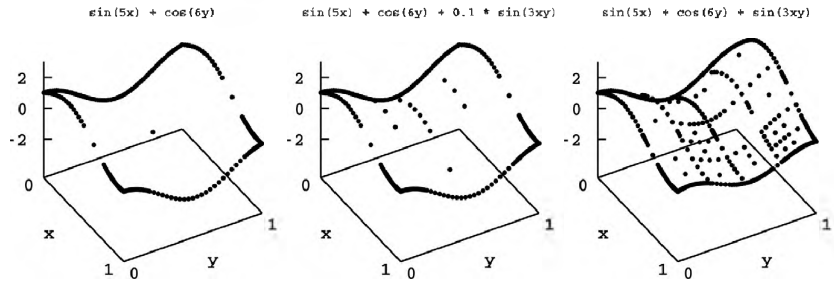

Рис. 103. Примеры интерполяции функций двух переменных

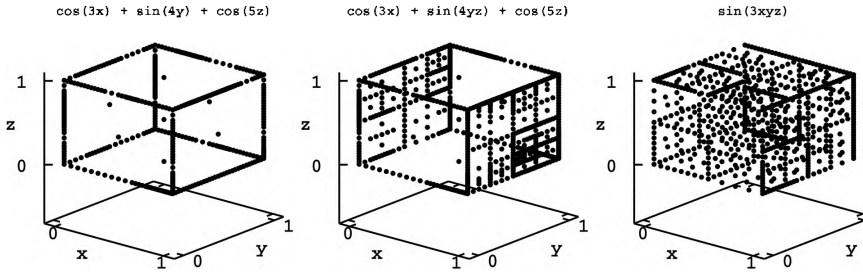

Рис. 104. Примеры сеток для функций трех переменных

Из рисунков видно, что данный подход определяет комбинации параметров, которые играют существенную роль, при этом построение сетки происходит не на всем множестве, а только на подмножествах, соответствующих этим комбинациям.

Рассмотрим систему ОДУ, описывающую модель Лотки–Вольтерры с тремя интервальными начальными условиями и семью интервальными параметрами:

$$\begin{cases} x' = x(\delta_1 - y - \varepsilon x), \\ y' = -\gamma_1 y(\delta_2 - x + z) - \varphi y^2, \\ z' = -\gamma_2 z(\alpha - y), \end{cases} \qquad \begin{cases} x(0),\ y(0),\ z(0),\ \delta_1,\ \delta_2,\ \gamma_1,\ \gamma_2 \in [1.0,\ 1.01], \\ \varepsilon,\ \varphi \in [-0.0005,\ 0.0005], \\ \alpha \in [0.9,\ 0.91]. \end{cases}$$

На рис. 105 показаны зависимости интервальных оценок решений от времени. Количество узлов в конечный момент времени равняется 305 481, апостериорная глобальная погрешность $\approx 10^{-2}$.

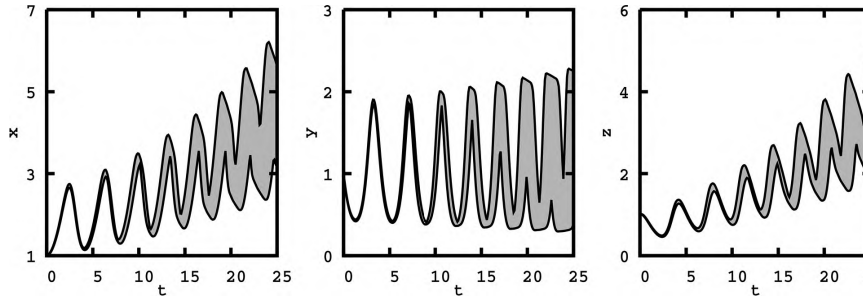

Рис. 105. Зависимость верхних и нижних оценок решения от времени

Как и в алгоритме адаптивной интерполяции, здесь происходит последовательное перемещение решения по временным слоям и перестроение адаптивной сетки для минимизации погрешности интерполяции.

5.5. ЗАКЛЮЧЕНИЕ

Описанные в главе подходы направлены на уменьшение экспоненциальной сложности при решении многомерных задач. Все они построены на предположении о том, что искомая функция имеет определенный вид. Какие-то параметры могут сильно влиять на результат, какие-то слабо, другие вообще могут никак не влиять. Автоматически учитывая данные особенности, можно эффективно уменьшить сложность решаемой задачи.

Использование TT-разложения в алгоритме адаптивной интерполяции позволяет расширить область его применения на случай большого числа интервальных параметров. Рассмотренные подходы успешно применены как к модельным задачам, так и к прикладным задачам химической кинетики.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Соболь И.М.* Метод Монте-Карло. — М.: Наука, 1978.
2. *Ермаков С.М., Михайлов Г.А.* Статистическое моделирование. — М.: Наука, 1982.
3. *Крамер Г.* Математические методы статистики. — М.: Мир, 1975.
4. *Золотарев В.М.* Общая теория перемножения независимых случайных величин // Доклады АН СССР. Т. 142. № 4. 1962. С. 788–791.
5. *Young R.C.* The algebra of many-valued quantities // Mathematische Annalen. Vol. 104. 1931. P. 260–290.
6. *Dwyer P.S.* Linear Computations. New York: John Wiley & Sons, 1951.
7. *Warmus M.* Calculus of Approximations // Bulletin de l'Academie Polonaise de Sciences. Vol. 4. № 5. 1956. P. 253–259.
8. Sunaga T. Theory of an Interval Algebra and its Application to Numerical Analysis // RAAG Memoirs. Vol. 2. 1958. P. 547–564.
9. *Moore R.E.* Interval Analysis. Englewood Cliffs: Prentice Hall, 1966.
10. *Lohner R.J.* Enclosing the solutions of ordinary initial and boundary value problems // Computer Arithmetic: Scientific Computation and Programming Languages. 1987. P. 255–286.
11. *Hansen E.* Interval Arithmetic in Matrix Computations Part I // SIAM Journal on Numerical Analysis. Vol. 2, №2. 1965. P. 308–320.
12. *Алефельд Г., Херцбергер Ю.* Введение в интервальные вычисления. — М.: Мир, 1987.
13. *Krawczyk R.* Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken // Computing. Vol. 4. №3. 1969. P. 187–201.
14. *Nickel K.* Über die Notwendigkeit einer Fehlerschranken-Arithmetic für Rechenautomaten // Numerische Mathematik. Vol. 9. № 1. 1966. P. 69–79.
15. *Neumaier A.* Interval Methods for Systems and Equations. Cambridge: Cambridge University Press, 1990.
16. *Брадис В.М.* Теория и практика вычислений. Пособие для высших педагогических учебных заведений. — М.: Учпедгиз, 1937.
17. *Канторович Л.В.* О некоторых новых подходах к вычислительным методам и обработке наблюдений // Сибирский математический журнал. Т. 3. № 5. 1962. С. 701–709.
18. *Калмыков С.А., Шокин Ю.И., Юлдашев З.Х.* Методы интервального анализа. — Новосибирск: Наука, 1986.
19. *Шокин Ю.И.* Интервальный анализ. — М.: Наука, 1981.
20. *Добронец Б.С., Попова О.А.* Численный вероятностный анализ неопределенных данных. — Красноярск: Сиб. федер. ун-т, 2014. — 168 с.
21. *Добронец Б.С.* Интервальная математика. — Красноярск: Краснояр. гос. ун-т., 2007.
22. *Шарый С.П.* Конечномерный интервальный анализ. — Новосибирск: XYZ, 2017.
23. *Рогалев А.Н.* Гарантированные методы решения систем обыкновенных дифференциальных уравнений на основе преобразования символьных формул // Вычислительные технологии. Т. 8. № 5. 2003. С. 102–116.

24. *Рогалев А.Н.* Вопросы реализации гарантированных методов включения выживающих траекторий управляемых систем // Сибирский журнал науки и технологий. № 2(35). 2011. С. 54–58.
25. *Рогалев А.Н.* Исследование и оценка решений обыкновенных дифференциальных уравнений интервально-символьными методами // Вычислительные технологии. Т. 4. № 4. 1999. С. 51–75.
26. *Морозов А.Ю., Ревизников Д.Л.* Модификация методов решения задачи Коши для систем обыкновенных дифференциальных уравнений с интервальными параметрами // Труды МАИ. № 89. 2016. С. 1–20.
27. *Eijgenraam P.* The Solution of Initial Value Problems Using Interval Arithmetic: Formulation and Analysis of an Algorithm. Amsterdam : Mathematisch Centrum, 1981. P. 185.
28. *Lohner R.J.* Einschließung der Losung gewohnlicher Anfangs– und Randwertaufgaben und Anwendungen. PhD thesis, Universitat Karlsruhe, 1988.
29. *Nedialkov N.S., Jackson K.R., Pryce J.D.* An effective high-order interval method for validating existence and uniqueness of the solution of an IVP for an ODE // Reliable Computing, Vol. 7. № 6. 2001. P. 449–465.
30. *Stauning O.* Automatic Validation of Numerical Solutions. PhD thesis, Technical University of Denmark, 1997.
31. *Lin Y., Stadtherr M.A.* Validated solutions of initial value problems for parametric ODEs. // Applied Numerical Mathematics. Vol 57. № 10. 2007. P. 1145–1162.
32. *Nedialkov N.S.* VNODE-LP — a validated solver for initial value problems in ordinary differential equations. Technical Report CAS-06-06-NN, Department of Computing and Software, McMaster University, 2006.
33. *Berz M., Makino K.* Verified integration of ODEs and flows with differential algebraic methods on Taylor models // Reliable Computing. Vol. 4. № 4. 1998. P. 361–369.
34. *Berz M., Makino K.* Suppression of the wrapping effect by Taylor model-based verified integrators: long-term stabilization by shrink wrapping // Differential Equations and Applications. Vol. 10. № 4. 2005. P. 385–403.
35. *Berz M., Makino K.* Suppression of the wrapping effect by Taylor model-based verified integrators: long-term stabilization by preconditioning // Differential Equations and Applications. Vol. 10. № 4. 2005. P. 353–384.
36. *Makino K., Berz M.* Efficient control of the dependency problem based on Taylor model methods // Reliable Computing. Vol. 5. № 1. 1999. P. 3–12.
37. *Makino K., Berz M.* Taylor models and other validated functional inclusion methods // Pure and Applied Mathematics Vol. 4. № 4. 2003. P. 379–456.
38. *Makino K., Berz M.* Verified Computations Using Taylor Models and Their Applications // Numerical Software Verification 2017: conference proceedings. (Heidelberg, Germany, July 22–23, 2017). Springer International Publishing AG 2017. P. 3–13.
39. *Neher M., Jackson K.R., Nedialkov N.S.* On Taylor Model Based Integration of ODEs // SIAM Journal on Numerical Analysis, Vol. 45. № 1. 2007. P. 236–262.
40. *Nataraj P.S.V., Sondur S.* The Extrapolated Taylor Model // Reliable Computing, Vol. 15. 2011. P. 251–278.
41. *Позин А.В.* Обзор методов и инструментальных средств решения задачи Коши для ОДУ с гарантированной оценкой погрешности // Международная

конференция «Современные проблемы прикладной математики и механики: теория, эксперимент и практика», 30 мая – 4 июня 2011 г. Новосибирск. Тезисы. ИВТ СО РАН, 2011.
42. *Berz M.* COSY INFINITY version 8 reference manual. Technical Report MSU-CL–1088, National Superconducting Cyclotron Lab., Michigan State Universitz, 1997.
43. *Eble I.* Über Taylor-Modelle: Dissertation zur erlangung des akademischen grades eines doktors der naturwissenschaften, Karlsruhe Institute of Technology, 2007.
44. *Chen X., Sankaranarayanan S.* Decomposed Reachability Analysis for Nonlinear Systems. // 2016 IEEE Real-Time Systems Symposium (RTSS): conference proceedings. (Porto, Portugal, 29 Nov.–2 Dec. 2016). P. 13–24.
45. *Chen X., Abraham E., Sankaranarayanan S.* FLOW*: An Analyzer for Non-linear Hybrid Systems // Proceedings of the 25th International Conference on Computer Aided Verification. (Saint Petersburg, Russia, July 13–19, 2013), Springer-Verlag New York, Vol. 8044, p. 258–263.
46. *Rump S.M.* INTLAB — INTerval LABoratory. In Tibor Csendes, editor, Developments in Reliable Computing, Kluwer Academic Publishers, Dordrecht, 1999, p. 77–104.
47. *Makino K., Berz M.* Rigorous Reachability Analysis and Domain Decomposition of Taylor Models // Numerical Software Verification 2017: conference proceedings. (Heidelberg, Germany, July 22–23, 2017). Springer International Publishing AG 2017, p. 90–97.
48. *Kletting M., Rauh A., Aschemann H., Hofer E.P.* Consistency tests in guaranteed simulation of nonlinear uncertain systems with application to an activated sludge process // Computational and Applied Mathematics. Vol. 199. №2. 2007. P. 213–219.
49. *Bünger F.* Shrink wrapping for Taylor models revisited // Numerical Algorithms. № 4. 2018. P. 1–18.
50. *Dobronets B.S.* On some two-sided methods for solving systems of ordinary differential equations // Interval Computation. 1992. Vol. 1. № 3. P. 6–19.
51. *Добронец Б.С., Рощина Е.Л.* Приложения интервального анализа чувствительности // Вычислительные технологии. Т. 7. № 1. 2002. С. 75–82.
52. *Некрасов С.А.* Эффективные двусторонние методы для решения задачи Коши в случае больших промежутков интегрирования // Дифференциальные уравнения. Т. 39. № 7. 2003. С. 969–973.
53. *Черноусько Ф.Л.* Оценивание фазовых состояний динамических систем. Метод эллипсоидов. — М.: Наука, 1988. — 319 с.
54. *Kurzhanski A.B., Vdlyi I.* Ellipsoidal Calculus for Estimation and Control. SCFA. Boston, 1997.
55. *Морозов А.Ю., Ревизников Д.Л.* Алгоритм адаптивной интерполяции на основе kd-дерева для численного интегрирования систем ОДУ с интервальными начальными условиями // Дифференциальные уравнения. Т. 54. № 7. 2018. С. 963–974.
56. *Морозов А.Ю., Ревизников Д.Л., Гидаспов В.Ю.* Алгоритм адаптивной интерполяции на основе kd-дерева для решения задач химической кинетики с интервальными параметрами // Математическое моделирование. Т. 30. №12. 2018. С. 129–144.

57. *Morozov A.Yu.*, *Reviznikov D.L.* Modelling of dynamic systems with interval parameters on graphic processors // Программная инженерия. Т. 10. 2. 2019. С. 69–76.
58. *Морозов А.Ю.*, *Ревизников Д.Л.*, *Гидаспов В.Ю.* Применение адаптивной интерполяции в задачах моделирования динамических систем с интервальными параметрами // Материалы XX Юбилейной международной конференции по вычислительной механике и современным прикладным системам (ВМСППС'2017). М.: МАИ-Принт, 2017. 816 с. С. 90–92.
59. *Морозов А.Ю.*, *Ревизников Д.Л.*, *Гидаспов В.Ю.* Алгоритм адаптивной интерполяции на основе kd-дерева для моделирования химических неравновесных течений с неопределенностями в константах скоростей реакций // Материалы XII Международной конференции по прикладной математике и механике в аэрокосмической отрасли (NPNJ'2018), 24–31 мая 2018 г., Алушта. — М.: Изд-во МАИ, 2018. С. 66–68.
60. *Сергеева Т.С.*, *Морозов А.Ю.* Использование графических процессоров в задачах интегрирования систем ОДУ с интервальными начальными условиями // Материалы XII Международной конференции по прикладной математике и механике в аэрокосмической отрасли (NPNJ'2018), 24–31 мая 2018 г., Алушта. — М.: Изд-во МАИ, 2018. С. 659–661.
61. *Морозов А.Ю.*, *Ревизников Д.Л.* Алгоритм адаптивной интерполяции для моделирования динамических систем с интервальными параметрами // Материалы XXII Междунар. науч.-практ. конф., посвящ. памяти генерального конструктора ракетно-космических систем академика М.Ф. Решетнева (12–16 нояб. 2018, г. Красноярск): в 2 ч. / под общ. ред. Ю.Ю. Логинова, СибГУ им. М.Ф. Решетнева. Красноярск, 2018. Ч. 2. С. 143–145.
62. *Морозов А.Ю.* Программа для численного интегрирования систем обыкновенных дифференциальных уравнений с интервальными начальными условиями // Свидетельство о государственной регистрации программы для ЭВМ №2018664623 от 20 ноября 2018 г.
63. *Berger M.J.*, *Colella*, *P.* Local adaptive mesh refinement for shock hydrodynamics // J. Comput. Phys. 1989. Vol. 82, Issue 1. P. 64–84.
64. *Мартыненко С.И.* Многосеточная технология: теория и приложения. — М.: Физматлит, 2015. — 170 с.
65. *Павлов Б.М.*, *Новиков М.Д.* Автоматизированный практикум по нелинейной динамике (синергетике). — М.: Диалог МГУ, ВМК, 2000. — 115 с.
66. *Красильников П.С.* Прикладные методы исследования нелинейных колебаний. — М.- Ижевск: Институт компьютерных исследований, 2015. – 528 с.
67. *Рыбаков К.А.*, *Рыбин В.В.* Моделирование распределенных и дробно-распределенных процессов и систем управления спектральным методом. — М.: Изд-во МАИ, 2016. — 160 с.
68. *Пантелеев А.В.*, *Рыбаков К.А.* Прикладной вероятностный анализ нелинейных систем управления спектральным методом. — М.: Изд-во МАИ-ПРИНТ, 2010. — 160 с.
69. *Bentley J.L.* Multidimensional binary search trees used for associative searching // Communications of the ACM. Vol. 18. № 9. 1975. P. 509–517.
70. *Russell A.B.* Building a Balanced k-d Tree in O(kn log n) Time // Computer Graphics Techniques. Vol. 4. № 1. 2015. P. 50–68.

71. *Казённов А.М.* Основы технологии CUDA // Компьютерные исследования и моделирование. Т. 2. № 3. 2010. С. 295–308.
72. *Hoefkens J., Berz M., Makino K.* Controlling the Wrapping Effect in the Solution of ODEs for Asteroids // Reliable Computing. Vol. 8. № 1. 2003. P. 21–41.
73. *Sauer T., Xu Y.* On multivariate lagrange interpolation // Mathematics of Computation. Vol. 64. № 211. 1995. P. 1147–1170.
74. *Гилл Ф., Мюррей У., Райт М.* Практическая оптимизация / Пер. с англ. — М.: Мир, 1985.
75. *Hansen E., Walster G.W.* Global Optimization Using Interval Analysis. New York: Marcel Dekker, 2004.
76. *Panteleev A.V., Panovskiy V.N.* Interval methods of global constrained optimization. Interval Analysis: Introduction, Methods and Applications. Nova Science Publishers, Inc. 2017. С. 33–119.
77. *Пантелеев А.В., Пановский В.Н.* Обобщенный инверсный интервальный метод глобальной условной оптимизации // Научный вестник Московского государственного технического университета гражданской авиации. №207. 2014. С. 17–24.
78. *Федорюк М.В.* Обыкновенные дифференциальные уравнения. — М.: Наука, 1980. — 352 с.
79. *Формалев В.Ф., Ревизников Д.Л.* Численные методы. — М.: Физматлит, 2004. — 400 с.
80. *Niesen J., Hall T.* On the Global Error of Discretization Methods for Ordinary Differential Equations // Ph.D. Thesis, University of Cambridge, 2004.
81. *Арнольд В.И.* Обыкновенные дифференциальные уравнения. Изд. 4-е. — Ижевск: Ижевская республиканская типография, 2000.
82. *Семенов С.А., Ревизников Д.Л.* Эффективное использование программируемых графических процессоров в задачах молекулярно-динамического моделирования // Системы и средства информатики. Т. 27. № 4. 2017. С. 109–121.
83. *Ревизников Д.Л., Семенов С.А.* Особенности молекулярно-динамического моделирования наносистем на графических процессорах // Программная инженерия. № 2. — М.: Новые технологии, 2013. С. 31–35.
84. *Dae-Hwan K.* Evaluation of the performance of GPU global memory coalescing // Multidisciplinary Engineering Science and Technology (JMEST). Vol. 4, № 4. 2017. P. 7009–7013.
85. *Sengupta S., Harris M., Garland M.* Efficient Parallel Scan Algorithms for GPUs: NVIDIA Technical Report, 2008.
86. *Alcantara D.A., Sharf A., Abbasinejad F., Sengupta S. et al.* Real-time Parallel Hashing on the GPU // ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH, Asia, 2009). Vol. 28, № 5. 2009. P. 1–9.
87. *Mei G., Tian H.* Impact of data layouts on the efficiency of GPU-accelerated IDW interpolation // SpringerPlus. Vol. 5. 2016. P. 1–18.
88. *Dinesh P.M., Sartaj S.* Handbook of Data Structures and Applications. Chapman & Hall/CRC, 2018, p. 1100.
89. *Роджерс Д., Адамс Дж.* Математические основы машинной графики. — М.: Мир, 2001. — 604 с.
90. *Соколов Ю.Н., Соколов А.Ю., Илюшко В.М.* Компьютерные технологии в задачах природы и общества. Ч. 2. Модель Лотки–Вольтерра «хищник–жертва»

в задачах экономики // Радіоелектронні і комп'ютерні системи. № 2 (43). 2010. С. 20–26.
91. *Neher M.* Interval methods and Taylor model methods for ODEs // Workshop Taylor Model Methods VII, 14–17 December 2011 y. Florida. Abstracts. MSU 2011. P. 17.
92. *Makino K., Berz M.* Suppression of the wrapping effect by Taylor model – based validated integrators: MSU HEP Report 40910, 2003.
93. *Арнольд В.И., Афраймович В.С., Ильяшенко Ю.С., Шильников Л.П.* Теория бифуркаций. — М.: ВИНИТИ, 1986. — 284 с.
94. *Красников С.Д., Кузнецов Е.Б.* Метод прохождения точек бифуркации коразмерности три // Прикладная математика и механика (Ульяновск). № 9. 2011. С. 335–346.
95. *Кузнецов Е.Б., Леонов С.С.* Параметризация задачи Коши для систем обыкновенных дифференциальных уравнений с предельными особыми точками // Журнал вычислительной математики и математической физики. Т. 57. № 6. 2017. С. 934–957.
96. *Кузнецов А.П., Кузнецов С.П., Рыскин Н.М.* Нелинейные колебания: Учебное пособие для вузов. — М.: Физматлит, 2002. — 292 с.
97. *Астахов В.В., Коблянский С.А., Шабунин А.В.* Осциллятор Дуффинга: Учебное пособие для студентов вузов. — Саратов: СГУ, 2007. — 53 с.
98. *Лоскутов А.Ю.* Динамический хаос. Системы классической механики // Успехи физических наук. Т. 177. № 9. 2007. С. 989–1015.
99. *Кроновер Р.М.* Фракталы и хаос в динамических системах. Основы теории. — М.: Постмаркет, 2000. — 352 с.
100. *Кулиш С.М., Морозов А.Ю., Тыкоцкий В.В.* Проблема астероидной опасности. Компьютерная программа «Астероид» // Тезисы доклада в сб. материалов XXIII Международной научно-практической конференции «Предупреждение. Спасение. Помощь» (Россия, Химки, 28 марта 2013 г.). С. 210.
101. *Кулиш С.М., Морозов А.Ю., Тыкоцкий В.В.* Проблема астероидной опасности – точность расчетов и измерений // Тезисы доклада в сб. материалов XXIII Международной научно-практической конференции «Предупреждение. Спасение. Помощь» (Россия, Химки, 28 марта 2013 г.). С. 211–212.
102. *Вайтиев В.А., Мустафина С.А.* Поиск областей неопределенности кинетических параметров математических моделей химической кинетики на основе интервальных вычислений // Вестник ЮУрГУ. Серия «Математическое моделирование и программирование». Т. 7. № 2. 2014. С. 99–110.
103. *Гидаспов В.Ю., Северина Н.С.* Элементарные модели и вычислительные алгоритмы физической газовой динамики. Термодинамика и химическая кинетика: Учебное пособие. — М.: Факториал, 2014. — 84 с.
104. *Глушко В.П., Гурвич Л.В., Вейц И.В. и др.* Термодинамические свойства индивидуальных веществ. Т. 1. — М.: Наука, 1978.
105. *Варнатц Ю., Маас У., Диббл Р.* Горение. Физические и химические аспекты, моделирование, эксперименты, образование загрязняющих веществ / Пер. с англ. Г.Л. Агафонова. Под ред. П.А. Власова. — М.: Физматлит, 2003. — 352 с.
106. *Старик А.М., Титова Н.С., Шарипов А.С., Козлов В.Е.* О механизме окисления синтез-газа // Физика горения и взрыва. Т. 46. № 5. 2010. С. 3–19.

107. *Новиков Е.А.*, *Голушко М.И.* (m, 3)-метод третьего порядка для жестких неавтономных систем ОДУ // Вычислительные технологии. Т. 3. № 3. 1998. С. 48–54.
108. *Пирумов У.Г.*, *Росляков Г.С.* Газовая динамика сопел. — М.: Наука. Гл. ред. физ.-мат. лит., 1990. — 368 с.
109. *Черный Г.Г.* Газовая динамика. — М.: Наука. Гл. ред. физ.-мат. лит., 1988. — 424 с.
110. *Журавская Т.А.*, *Левин В.А.* Стабилизация детонационного горения высокоскоростного потока горючей газовой смеси в плоском канале // Изв. РАН. МЖГ. № 2. 2015. С. 117–128.
111. *Тыртышников Е.Е.* Тензорные аппроксимации матриц, порожденных асимптотически гладкими функциями // Математический сборник. 2003. Т. 194. № 6. С. 147–160.
112. *Желтков Д.А.*, *Тыртышников Е.Е.* Параллельная реализация матричного крестового метода // Вычислительные методы и программирование. 2015. Т. 16. С. 369–375.
113. *Горейнов С.А.*, *Замарашкин Н.Л.*, *Тыртышников Е.Е.* Псевдоскелетные аппроксимации при помощи подматриц наибольшего объема // Математические заметки. 1997. Т. 62. Вып. 4. С. 619–623.
114. *Hitchcock F.L.* The expression of a tensor or a polyadic as a sum of products // J. Math. Phys. 1927. Vol. 6, no. 1. P. 164–189.
115. *Tucker L.R.* Some mathematical notes on three-mode factor analysis // Psychometrika. 1966. Vol. 31. P. 279–311.
116. *Oseledets I.*, *Tyrtyshnikov E.* TT-cross approximation for multidimensional arrays // Linear Algebra and its Applications Vol. 432, Iss. 1. 2010. P. 70–88.
117. *Oseledets I.V.* Tensor-train decomposition // SIAM Journal on Scientific Computing 2011 Vol. 33, no. 5. P. 2295–2317.
118. *Смоляк С.А.* Квадратурные и интерполяционные формулы на тензорных произведениях некоторых классов функций, Докл. АН СССР, 148:5 (1963), 1042–1045.
119. *Gerstner T.*, *Griebel M.* Sparse Grids // Encyclopedia of Quantitative Finance, R. Cont (ed.), Wiley, 2008.
120. *Garcke J.* Sparse Grids in a Nutshell // Sparse Grids and Applications. Lecture Notes in Computational Science and Engineering. 2013. V. 88. Springer, Berlin, Heidelberg. P. 57–80.
121. *Brumm J.*, *Scheidegger S.* Using Adaptive Sparse Grids to Solve High-Dimensional Dynamic Models // Econometrica. 2017. Vol. 85, I. 5. P. 1575–1612.
122. *Bungatrz H-J.*, *Griebel M.* Sparse grids // Acta Numerica. 2004. Vol. 13, no. 1. P. 147–269.

ОГЛАВЛЕНИЕ

Научное издание

Морозов Александр Юрьевич
Ревизников Дмитрий Леонидович

МЕТОДЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ ДИНАМИЧЕСКИХ СИСТЕМ С ИНТЕРВАЛЬНЫМИ ПАРАМЕТРАМИ

Редактор *Е.В. Дмитриева*
Компьютерная верстка *О.А. Пелипенко*

Сдано в набор 14.10.2019. Подписано в печать 18.11.2019.
Бумага писчая. Формат 60×84 1/16. Печать офсетная.
Усл. печ. л. 9,30. Уч.-изд. л. 10,0. Тираж 500 экз.
Заказ 1066/730.

Издательство МАИ
(МАИ), Волоколамское ш., д. 4
Москва, А-80, ГСП-3 125993

Типография Издательства МАИ
(МАИ), Волоколамское ш., д. 4
Москва, А-80, ГСП-3 125993

НАУЧНАЯ БИБЛИОТЕКА