МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)

На правах рукописи

Пановский Валентин Николаевич

ИНТЕРВАЛЬНЫЕ МЕТОДЫ ОПТИМИЗАЦИИ НЕЛИНЕЙНЫХ ДЕТЕРМИНИРОВАННЫХ ДИНАМИЧЕСКИХ СИСТЕМ ПРИ НЕПОЛНОЙ ИНФОРМАЦИИ О СОСТОЯНИИ И ПАРАМЕТРАХ ОБЪЕКТА

Специальность 05.13.18
Математическое моделирование, численные методы и комплексы программ

Специальность 05.13.01
Системный анализ, управление и обработка информации
(авиационная и ракетно-космическая техника)

Диссертация на соискание ученой степени
кандидата физико-математических наук

Научный руководитель
доктор физико-математических наук,
профессор А.В. Пантелеев

Москва, 2017

ОГЛАВЛЕНИЕ

Введение 5

Глава 1. Разработка интервальных методов поиска глобального условного экстремума 18

1.1. Постановка задачи интервальной ε-минимизации 18

1.2. Инверсные методы решения задачи интервальной ε-минимизации 20

1.2.1. Метод дихотомии целевого интервала 21

1.2.2. Метод отсечки виртуальных значений 22

1.2.3. Метод стохастической отсечки виртуальных значений 24

1.2.4. Метод стохастических вырываний 26

1.2.5. Обобщенный инверсный метод 28

1.2.6. Теоремы о свойствах решений интервальной ε-минимизации инверсными методами 34

1.3. Метаэвристические методы решения задачи интервальной ε-минимизации 36

1.3.1. Метод усредненных концов путей 37

1.3.2. Метод стохастической сетки 39

1.3.3. Метод интервального разбросанного поиска 41

1.3.4. Интервальный генетический алгоритм 44

1.3.5. Интервальный метод взрывов 49

1.3.6. Адаптивный интервальный алгоритм 52

1.3.7. Самоорганизующийся интервальный алгоритм имитации эволюции колонии бактерий 59

1.4. Тестирование интервальных методов оптимизации 69

1.4.1. Метод дихотомии целевого интервала 72

1.4.2. Метод отсечки виртуальных значений 72

1.4.3. Метод стохастической отсечки виртуальных значений 73

1.4.4. Метод стохастических вырываний 73

1.4.5. Обобщенный инверсный метод 73

1.4.6. Метод усредненных концов путей 74
1.4.7. Метод стохастической сетки 74
1.4.8. Метод интервального разбросанного поиска 74
1.4.9. Интервальный генетический алгоритм 75
1.4.10. Интервальный метод взрывов 75
1.4.11. Адаптивный интервальный алгоритм 75
1.4.12. Самоорганизующийся интервальный алгоритм имитации эволюции колонии бактерий 76
1.5. Заключение 76
Глава 2. Интервальные алгоритмы синтеза оптимальных динамических систем 78
2.1. Интервальные алгоритмы нахождения оптимального программного управления нелинейными детерминированными динамическими системами 78
2.1.1. Постановка задачи 78
2.1.2. Стратегия поиска управления 79
2.1.3. Алгоритм поиска управления 81
2.2. Интервальные алгоритмы нахождения оптимального управления с неполной обратной связью нелинейными детерминированными динамическими системами 81
2.2.1. Постановка задачи 81
2.2.2. Стратегия поиска управления 83
2.2.3. Алгоритм поиска управления 85
2.3. Интервальные алгоритмы нахождения оптимального управления по выходу нелинейными детерминированными динамическими системами при неопределенности в параметрах модели объекта управления и модели измерений 86
2.3.1. Постановка задачи 86
2.3.2. Стратегия поиска управления 88
2.3.3. Алгоритм поиска управления 91
2.4. Заключение 92
Глава 3. Программный комплекс «Интервальные методы оптимизации нелинейных детерминированных систем» 93

Глава 4. Приложение интервальных методов в задачах оптимизации технических систем и управления авиационно-космическими системами 96
4.1. Задачи оптимизации технических систем 97
4.1.1. Задача определения параметров сварной балки 97
4.1.2. Задача определения параметров сосуда высокого давления 99
4.1.3. Задача определения параметров редуктора 100
4.1.4. Задача определения параметров натяжной/компрессионной пружины 102
4.2. Задачи оптимального управления авиационно-космическими системами 104
4.2.1. Задача преследования 104
4.2.2. Задача об управлении солнечным парусом 110
4.2.3. Задача о командной навигации 112
4.2.4. Задача о приземлении гиперзвукового летательного аппарата 114
4.2.5. Задача о стабилизации спутника 117
4.2.6. Задача о перехвате 120
4.3. Заключение 123
Заключение 125
Приложение. Введение в интервальный анализ 127
П.1. Основные понятия интервального анализа 127
П.1.1. Интервалы и интервальные векторы 127
П.1.2. Интервальные арифметики 128
П.1.3. Интервальное расширение функций 129
П.2. Инвертер 130
Библиографический список 132

ВВЕДЕНИЕ

Диссертационная работа посвящена разработке интервальных алгоритмов глобальной условной оптимизации для решения задач оптимального управления нелинейными детерминированными динамическими системами при неполной информации о состоянии и параметрах объекта и их применению в задачах авиационной и ракетно-космической техники.

Актуальность работы. В современной математике достаточно большое внимание уделяется решению задач глобальной оптимизации и синтеза оптимального управления динамическими системами. Эти задачи возникают в ходе проектирования конструкций самолетов, вертолетов, космических аппаратов, когда появляется необходимость оптимизации характерных параметров (вес, дальность полета, аэродинамические характеристики) и разработки систем управления как отдельными элементами конструкции, так и объектом в целом. В большинстве случаев для нахождения приближенного решения требуется использовать численные методы вследствие невозможности применения аналитических подходов, основанных на необходимых и достаточных условиях оптимальности. На сегодняшний день известны эффективные численные методы, разработанные Евтушенко Ю.Г., Моисеевым Н.Н., Крыловым И.А., Черноусько Ф.Л., Тихоновым А.Н., Васильевым Ф.П., Колмановским В.Б., Кротовым В.Ф., Гурманом В.И., Хрусталевым М.М., Федоренко Р.П., Bryson A.E., Ho Y-G, Пропоем А.И., Габасовым Р.Ф., Кирилловой Ф.М., Батуриным В.А., Срочко В.А., Дыхтой В.А., Васильевым С.Н., Levine W.S, Hellerstein J.L., Tilbury D.M. и др.

В последнее время стала более значимой роль метаэвристических алгоритмов оптимизации. Несмотря на отсутствие строгого обоснования, эти алгоритмы позволяют найти приемлемое решение задачи в большинстве практически значимых случаев (не обязательно наилучшее). Достоинством таких алгоритмов является их относительно низкая вычислительная сложность, что позволяет применять их для решения задач повышенной трудности. Описание данных алгоритмов можно найти в работах Glover F., Laguna M., Marti R., Gendreau M., Moscato P., Cotta C., Dorigo M., Yang X-S., Holland J, Пантелеева А.В., Метлицкой Д.В., Алешиной Е.А., Карпенко А.П., Курейчика В.М. и др.

Вследствие того, что оптимизируемые математические модели постоянно усложняются, необходимо пробовать новые подходы при разработке численных методов с целью создания вычислительно более эффективных алгоритмов оптимизации. Требуется учитывать неопределенности задания начальных условий, параметров моделей объектов

управления, неполноту и неточность информации, получаемой от измерительных устройств. Во многих практических задачах характерные параметры задаются векторами, компоненты которых определяются интервалами их возможного изменения. В связи с вышеприведенными утверждениями применение **интервального анализа** в качестве базового элемента описания, анализа и численных методов оптимизации является достаточно естественным. Свое развитие эта математическая дисциплина получила в XX веке совместно с распространением практических вычислений. Эволюция интервального анализа и его оформление в виде самостоятельной научной дисциплины напрямую связано с появлением компьютеров. В XX веке произошло несколько важных событий, предшествующих появлению интервального анализа. Среди них:

- разработка арифметики для вычислений с множествами чисел англичанкой Р. Янг в 1931 году [124],
- рассмотрение специального случая замкнутых интервалов для учета погрешностей в численном анализе П. Двайером в 1951 году [87],
- появление работ М. Вармуса в Польше в 1956 году [123] и Т. Сунаги в Японии в 1958 году [120], в которых предлагалась классическая интервальная арифметика и ее приложения.

В 1962 году Р. Мур написал свою докторскую диссертацию по использованию интервальных оценок при анализе ошибок вычислений и контроля за ними на компьютерах. В 1966 году он написал первую систематическую монографию по интервальному анализу [105]. Примерно в то же время Э. Хансен изучал операции с интервалами в линейной алгебре [92], а группа немецких исследователей (в том числе Г. Алефельд, Р. Кравчик и К. Никель) развила многие вопросы компьютерной реализации методов [1, 109]. Кроме того, следует отметить работы А. Ноймайера в области прикладного применения интервального анализа [125].

В России и Советском Союзе история интервального анализа отсчитывается с 20-х годов прошлого века и связана с именем русского математика В.М. Брадиса и его методом границ – способом организации вычислений, приводящим к достоверным двусторонним границам точного значения вычисляемого результата, что по сути является аналогом интервальной арифметики [2]. В 1962 году появилась статья Л.В. Канторовича [10], который обозначает эту тематику как приоритетную для вычислительной науки.

Кроме того, необходимо упомянуть и других ученых, внесших существенный вклад в развитие интервального анализа: Н.Н. Яненко, Ю.И. Шокина (написавшего первую книгу по интервальному анализу на русском языке) [9, 79], Б.С. Добронца [5].

Следует отметить работы С.П. Шарого [73–77, 117] и его ученика Н.В. Панова [17, 18], занимающиеся исследованиями в различных областях применения интервального анализа и оптимизации в частности, а также работы Г.Г. Меньшикова [12–15]. В их работах можно найти подробное описание интервальных алгоритмов, использующих различные подходы при решении поставленных задач.

На сегодняшний день аппарат интервального анализа очень сильно расширился.

Имеется большое количество интервальных арифметик [76, 99], каждая из которых имеет свои особенности, которые делают ее более подходящей для решения конкретного круга задач:

- классическая интервальная арифметика – базовая арифметика, определяющая основные операции над интервалами,
- полная интервальная арифметика (интервальная арифметика Каухера) – арифметика, вводящая понятия неправильных интервалов, т.е. интервалов, правая граница которых может быть меньше, чем левая,
- комплексные интервальные арифметики – интервальный вариант комплексных чисел,
- твинная арифметика – арифметика двойных интервалов (интервалов с интервальными концами),
- арифметика Кахана – интервальная арифметика, позволяющая производить деление на нульсодержащий интервал,
- мультиинтервальная арифметика – арифметика над объединениями конечного числа несвязных интервалов на числовой оси,
- сегментные арифметики – аналог интервальной арифметики, определенной над множествами сегментов.

Существующие методы позволяют производить интервальное оценивание областей значений вещественных функций. Наиболее значимые результаты в этой области:

- теоремы о внешних оценках области значения функций, полученные Р. Муром [105],
- идеи, основанные на использовании «остаточных форм функции f», предложенные Х. Корнелиусом и Р. Лонером [85],
- применение обобщенных центрированных форм, описанных Г.Алефельдом в [80].

В области решения систем нелинейных уравнений вида $f(x)=0$ и $f(x)=\chi$ крайне важными являются следующие результаты:

- методы локализации нулей заданной функции f, основанные на результате К. Миранды [104],

- метод проб и ошибок С. Румпа [115],
- оператор Р. Кравчика [100].

Для решения систем линейных уравнений используются следующие методы:

- интервальный метод Холесского [81],
- метод внешнего оценивания симметричного множества решений К. Янссона [96].

Кроме перечисленных задач, интервальные методы позволяют решать алгебраические проблемы собственных значений и векторов:

- теорема С. Румпа о существовании неособенной интервальной матрицы [96],
- методы определения внешних интервальных оценок собственных значений и векторов Г. Бенке и Ф. Гериша [83] и Р. Лонера [102].

В области обыкновенных дифференциальных уравнений большая часть работ связана с доказательными численными методами решения задачи Коши. Среди этих работ можно выделить следующие:

- интервальный метод, использующий разложения в ряды Тейлора, предложенный Р. Муром [88],
- интервальный метод Эрмита-Обрешкова [108].

Наиболее значимые результаты в области решения дифференциальных уравнений в частных производных:

- метод М. Плюма для решения краевых задач эллиптического типа [112],
- идеи М. Накао для решения задачи Дирихле [107].

Приведем обзор существующих **интервальных методов оптимизации**. Рассмотрим существующие интервальные методы поиска глобального минимума функции. Их можно разделить на две группы в зависимости от решаемой задачи: методы условной и безусловной оптимизации.

К методам безусловной оптимизации можно отнести следующие методы:

- Интервальный адаптивный алгоритм, или алгоритм Мура-Скелбоу [114]. Алгоритм оперирует с рабочим списком L, в котором хранятся интервальные векторы, получающиеся в результате дробления исходного интервального вектора на более мелкие. Кроме этих интервальных векторов в списке хранятся и нижние оценки областей значений целевой функции на них, что приводит к усложнению реализации алгоритма. На каждом шаге алгоритма из списка извлекается вектор с наименьшей оценкой значения функции, производится его дробление, оценка получившихся интервальных векторов и занесение результатов обратно в список.
    - Достоинства: простота реализации.

- Недостатки: метод использует лишь базовые понятия интервального анализа, большое количество переключений между анализируемыми интервальными векторами.

- Алгоритм Ичиды-Фуджи [114]. По сути является модификацией алгоритма Мура-Скелбоу, добавляя в него стадию модификации рабочего списка L.
  - Достоинства: простота реализации.
  - Недостатки: метод использует лишь базовые понятия интервального анализа, большое количество переключений между анализируемыми интервальными векторами.
- Метод Дюсселя [86]. Основная идея данного метода заключаются в следующем: избегать использования списков (взамен предлагается использовать рекурсивные процедуры).
  - Достоинства: отсутствие необходимости работать со списками.
  - Недостатки: средняя сложность реализации.
- Метод Хансена [114], который имеет вариации, использующие градиент функции, понятие выпуклости и использование пробных точек. Оригинальный метод, так же как и методы Мура-Скелбоу и Ичиды-Фуджи, работает со списком, в котором хранятся интервальные векторы и оценка значения функции на них.
  - Достоинства: более высокая скорость сходимости при использовании аппарата математического анализа, простота реализации оригинального метода.
  - Недостатки: большая вычислительная сложность вариаций метода (в связи с необходимостью реализации численного дифференцирования или проверки выпуклости и т.п.), высокая сложность реализации.
- Интервальный метод Ньютона [93].
  - Достоинства: более высокая скорость сходимости за счет использования производной.
  - Недостатки: большая вычислительная сложность вариаций метода (в связи с необходимостью реализации численного дифференцирования).
- Интервальный алгоритм «имитации отжига», разработанный С.П. Шарым [117]. В основе метода лежит классический (точечный) метод, который моделирует физические процессы отжига или кристаллизации.
  - Достоинства: использует в основе хорошо изученный алгоритм, который уже показал высокую эффективность.

- Недостатки: так как метод относится к классу метаэвристических – его достаточно сложно настраивать, для более точной работы требуется бесконечно медленное охлаждение, что ведет к увеличению времени работы алгоритма.

- Метод случайного интервального дробления, разработанный С.П. Шарым [117]. Данный метод можно назвать модификацией интервального адаптивного алгоритма. Главное отличие в том, что интервальный вектор выбирается из рабочего списка L случайным образом.
  - Достоинства: простота реализации.
  - Недостатки: метод использует лишь базовые понятия интервального анализа, большое количество переключений между анализируемыми интервальными векторами.
- Метод дробления графика, разработанный С.П. Шарым [74]. Данный метод тоже похож на интервальный адаптивный алгоритм, а основное новшество заключается в том, что метод ищет оценку минимального значения функции $f(x)$ за счет анализа совместности уравнения $f(x)-l=0$, где l – некоторое значение, которое функция может принимать. С помощью такого анализа можно определить больше или меньше, чем l, значение минимума функции.
  - Достоинства: простота реализации.
  - Недостатки: необходимость корректной обработки списка (чтобы не нарушить упорядоченность), что сопряжено с увеличением вычислительной сложности.
- Интервальный эволюционный алгоритм, разработанный Н.В. Пановым и С.П. Шарым [18]. Данный метод похож на интервальный адаптивный алгоритм, модифицированный некоторыми понятиями эволюционных алгоритмов (в частности, оператором мутации).
  - Достоинства: использует в основе хорошо изученный алгоритм, который уже показал высокую эффективность.
  - Недостатки: так как метод относится к классу метаэвристических – его достаточно сложно настраивать, высокие вычислительная трудоемкость и сложность реализации.

К методам условной оптимизации можно отнести следующие интервальные методы:

- Метод Хансена [93], который использует условия Фрица-Джона и численные методы решения систем. Применение данных условий позволяет оптимизировать целевую функцию при ограничениях обоих типов (и равенств, и неравенств).
  - Достоинства: теоретическая обоснованность метода.

- Недостатки: большая вычислительная сложность метода (в связи с необходимостью реализации численного дифференцирования), высокая сложность реализации.

- Метод Мура [106], который использует условия Куна-Таккера-Каруша. Использование данных условий позволяет оптимизировать целевую функцию при ограничениях типа неравенств.
  - Достоинства: теоретическая обоснованность метода.
  - Недостатки: большая вычислительная сложность метода (в связи с необходимостью реализации численного дифференцирования), высокая сложность реализации.

Перечисленные методы имеют ряд недостатков, связанных прежде всего с требованием дифференцируемости и выпуклости целевой функции и функций, описывающих ограничения. Компьютерная реализация данных процедур является достаточно сложной и дополнительно увеличивает вычислительные затраты метода. Кроме того, большинство методов так или иначе оперируют с рабочим списком. Ведение и обработка этого списка также увеличивает вычислительную сложность алгоритма, так как требуется проводить много операций сравнения, чтобы не нарушить упорядоченность элементов.

Теория управления. Основной задачей теории оптимального управления является задача проектирования системы, которая обеспечит для заданного объекта управления или процесса закон управления или управляющую последовательность воздействий, обеспечивающих максимум или минимум заданного критерия качества системы [67].

Для решения задачи оптимального управления строится математическая модель управляемого объекта или процесса, описывающая его поведение с течением времени под влиянием управляющих воздействий и собственного текущего состояния. Математическая модель для задачи оптимального управления, как правило, включает в себя: формулировку цели управления, выраженную через критерий качества управления; определение дифференциальных или разностных уравнений, описывающих движение объекта управления; определение ограничений на используемые ресурсы в виде уравнений или неравенств [57].

Основу классической теории оптимального управления составляют следующие методы:

- методы вариационного исчисления,
- принцип максимума Понтрягина,

- динамическое программирование Беллмана,
- условия глобальной оптимальности Кротова.

Сама суть реализации любого управления на практике сопряжена с информационной неопределенностью. Это является следствием того, что параметры модели известны неточно (задаются интервалами неопределенности), а любая получаемая информация о поведении объекта искажается ошибками измерений, которые в свою очередь следуют из неточности приборов. Именно поэтому использование интервального анализа как базовой составляющей методов поиска оптимального управления является естественным и перспективным.

Условно интервальные методы, применяемые в теории управления, можно разделить на следующие группы:

- методы, основанные на применении аппарата функций чувствительности и частотном представлении объекта:
    - работа В.Н. Ефанова, В.Г. Крымского и Р.З. Тляшова на тему синтеза многосвязных систем с интервальными характеристическими полиномами [6],
    - работа Т. Гарденеса и А. Трипата на тему интервальных вычислительных систем [90] (использование понятий интервального анализа для аппарата функций чувствительности),
- методы синтеза робастных систем:
    - работа В.Л. Харитонова о положении равновесия семейства систем линейных дифференциальных уравнений [70] (решение задачи об асимптотической устойчивости интервального характеристического полинома),
    - работа Е.М. Смагиной и И. Брюйера о применении интервальной арифметики для создания робастной обратной связи [119] (доказана связь между управляемостью интервальной системой и существованием робастного модального регулятора),
    - работа М.В. Морозова об условиях робастной устойчивости при интервальных ограничениях [16] (для случая периодических интервальных ограничений найден вид систем, для которых достаточные условия робастной устойчивости систем являются необходимыми),
    - работа Ю.М. Гусева, В.Н. Ефанова, В.Г. Крымского и В.Ю. Рутковского о синтезе линейных интервальных динамических систем [3, 4] (проводится обзор результатов, относящихся к анализу линейных интервальных динамических систем, на основании информации об их характеристических полиномах или информации об элементах матриц, участвующих в записи уравнений их состояния),

- работа В.Н. Шашихина о методах интервального анализа в синтезе робастного управления [78] (в работе предложен метод решения интервальных матричных уравнений, который основывается на итеративной процедуре решения двух алгебраических уравнений с вещественными коэффициентами),

- адаптивные методы:
  - работа Е.М. Смагиной и И.В. Дугаровой о модальном регуляторе для систем с неопределенными параметрами [68] (предложены алгоритмы синтеза управления для многосвязных линейных интервальных динамических систем),

- методы модального управления:
  - работа А.В. Захарова и Ю.И. Шокина о синтезе систем управления при интервальной неопределенности параметров [7] (методе синтеза оптимальных робастных регуляторов многомерной линейной стационарной системой с интервальными матрицами),
  - работа Н.А. Хлебалина о синтезе интервальных регуляторов [72] (приведены условия разрешимости задачи модального синтеза систем управления с интервальными параметрами),
  - работа Н.А. Хлебалина по синтезу регуляторов в условиях неопределенности параметров объекта [71] (в работе описаны процесс синтеза устойчивого интервального полинома, критерии управляемости объектов с неопределенными параметрами и синтез регуляторов интервальным вариантом метода модального управления),
  - работа Р.С. Ивлева и С.П. Соколовой на тему построения управления интервально заданным объектом [8] (в работе решается задача параметрического синтеза путем определения интервальной матрицы настраиваемых параметров алгоритма управления),
  - работа С.П. Шарого о методах решения линейной задачи о допусках [73] (рассматривается метод поиска интервального вектора, содержащегося в допустимом множестве решений рассматриваемой интервальной линейной системы уравнений),
  - работа Е.М. Смагиной и А.Н. Моисеева о слежении за полиномиальным сигналом в интервальной динамической системе [69] (представлен метод, приводящий проблему слежения за полиномиальным сигналом в динамической системе к синтезу модального регулятора для расширенной системы).

Таким образом, диссертационная работа посвящена разработке интервальных алгоритмов глобальной условной оптимизации и их применению для решения задач поиска оптимального управления нелинейными непрерывными детерминированными динамическими системами при неполной информации о состоянии и параметрах объекта.

Целью работы является разработка алгоритмического и программного обеспечения интервальных алгоритмов глобальной условной оптимизации для решения задач поиска оптимального управления нелинейными детерминированными динамическими системами. В диссертации были поставлены и решены следующие задачи:

1) разработка интервальных алгоритмов оптимизации на основе инвертера (процедуры поиска прообраза множества значений функции),
2) разработка метаэвристических интервальных алгоритмов оптимизации,
3) разработка интервальных алгоритмов поиска оптимального программного управления,
4) разработка интервальных алгоритмов синтеза оптимального в среднем управления пучками траекторий с неполной обратной связью и управления по выходу,
5) разработка комплекса программ, включающего интервальные методы оптимизации и интервальные методы синтеза оптимального управления,
6) применение разработанного алгоритмического и программного обеспечения для решения задач оптимизации технических систем и управления авиационно-космическими системами.

Методы исследования. Для исследования теоретических вопросов использовались интервальный анализ, теория оптимизации, численные методы, теория управления, математическая статистика.

Научная новизна. В диссертационной работе получены новые результаты: разработаны интервальные алгоритмы глобальной условной оптимизации двух типов (основывающиеся на инвертере и метаэвристические), которые были применены для решения задач оптимизации технических систем, а также поиска оптимального программного управления, оптимального в среднем управления пучками траекторий с неполной обратной связью и оптимального в среднем управления по выходу нелинейными детерминированными динамическими системами.

Практическая значимость. В диссертационной работе разработаны приближенные интервальные методы решения задач оптимального управления нелинейными детерминированными динамическими системами с неопределенностью параметров и начальных условий, которые применимы в области авиационной и ракетно-космической техники. Создан комплекс программ для решения прикладных задач поиска оптимального

управления при помощи интервальных алгоритмов глобальной условной оптимизации. Была произведена государственная регистрация разработанных программ (свидетельства №2015661635, №2016610641), с помощью которых были решены прикладные задачи оптимизации технических систем и управления авиационно-космическими системами.

Достоверность результатов. Эффективность интервальных алгоритмов глобальной условной оптимизации была проверена на наборе тестовых функций, для которых известно точное решение, а также на прикладных задачах теории управления, для которых известно решение, найденное другими приближенными методами. Приведенные в диссертационной работе результаты не противоречат уже известным решениям. Полученные приближенные решения прикладных задач полностью отвечают физической картине мира.

Апробация работы. Результаты диссертационной работы докладывались на следующих научных конференциях: Международная конференция «Авиация и космонавтика» (Москва, 2010 – 2013), Всероссийская научно-техническая конференция «Прикладные научно-технические проблемы современной теории управления системами и процессами» (Москва, 2012), 15th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetics and Verified Numerics (Новосибирск, 2012), Международная конференция «Инжиниринг & Телекоммуникации – EN&T 2015» (Москва/Долгопрудный, 2015), НТК молодых ученых и специалистов ПАО «НПО «Алмаз» по тематике «Актуальные вопросы развития систем и средство ВКО» (Москва, 2013 – 2016). Результаты диссертационного исследования были отмечены на конференциях, посвященных информационным технологиям (на научно-технической конференции «Актуальные вопросы развития систем и средств ВКО» в секции «Информационные технологии. Автоматизированные системы управлений войсками и оружием» представленные работы занимали I место в 2013 и 2016 году, II место в 2015 году). Исследования были поддержаны грантами РФФИ (гранты № 16-07-00419-а, № 16-31-00115-мол_а).

Публикации. Основные результаты диссертационной работы опубликованы в статьях [27, 42–44, 48, 55, 56, 62–64] в журналах, входящих в Перечень ВАК, в других изданиях [19, 20, 24, 28, 29, 59–61, 111] и в трудах научных конференций [21–23, 25, 26, 30–41, 45–47, 49–51, 53, 58, 110]. Получены 2 свидетельства о государственной регистрации программ [52, 54]. Всего по теме диссертации опубликовано 47 работ.

Структура и объем диссертации. Диссертационная работа состоит из введения, четырех глав основной части, заключения, списка использованных источников (125 наименований) и одного приложения. Работа изложена на 139 страницах и содержит 49 иллюстраций и 26 таблиц.

Основным итогом диссертационной работы является разработка интервальных методов глобальной условной оптимизации и их применение для решения задач оптимизации технических систем и поиска оптимального управления (программного, с неполной обратной связью и управления по выходу) нелинейными детерминированными динамическими системами при неполной информации о состоянии и параметрах объекта, выразившиеся в следующих основных результатах:

1) разработаны интервальные алгоритмы оптимизации на основе инвертера (методы дихотомии целевого интервала, отсечки виртуальных значений, стохастической отсечки виртуальных значений, стохастических вырываний, обобщенный инверсный метод) [27, 42, 63],
2) разработаны интервальные метаэвристические алгоритмы оптимизации (среднего пути, стохастической сетки, интервального разбросанного поиска, интервальный генетический алгоритм, интервальный метод взрывов, адаптивный интервальный алгоритм, самоорганизующийся интервальный алгоритм имитации эволюции колонии бактерий) [43, 44, 48, 55, 56, 62, 64, 111],
3) разработаны интервальные алгоритмы поиска оптимального программного управления нелинейными непрерывными детерминированными системами [27, 42–44, 48, 55, 62–64, 111],
4) разработаны интервальные алгоритмы синтеза оптимального в среднем управления с неполной обратной связью и управления по выходу нелинейными непрерывными детерминированными системами [56],
5) разработан программный комплекс, реализующий интервальные методы глобальной условной оптимизации и алгоритмы их применения для поиска оптимального программного управления, оптимального в среднем управления пучками траекторий с неполной обратной связью и оптимального управления по выходу при неполной информации о состоянии и параметрах объекта [52, 54],
6) разработанное алгоритмическое и программное обеспечение применено для решения задач оптимизации технических систем (определение оптимальных параметров сварной балки, сосуда высокого давления, редуктора, натяжной/компрессионной пружины) и систем управления авиационно-космической техникой (задачи о преследовании, управлении солнечным парусом, командной навигации, стабилизации спутника, перехвате, управлении гиперзвуковым летательным аппаратом) [27, 42–44, 48, 55, 56, 62–64, 111].

Диссертационная работа соответствует паспорту специальности 05.13.18 (в работе проведены разработка, обоснование и тестирование эффективных численных методов с применением современных компьютерных технологий; реализация эффективных численных методов и алгоритмов в виде комплексов проблемно-ориентированных программ для проведения вычислительного эксперимента; разработка систем компьютерного и имитационного моделирования; предложена математическая модель интервальной задачи ε-минимизации) и паспорту специальности 05.13.01 (проведена разработка специального математического и алгоритмического обеспечения систем оптимизации и управления).

ГЛАВА 1. РАЗРАБОТКА ИНТЕРВАЛЬНЫХ МЕТОДОВ ПОИСКА ГЛОБАЛЬНОГО УСЛОВНОГО ЭКСТРЕМУМА

В данной главе разработаны интервальные алгоритмы глобальной условной оптимизации двух типов: на основе инвертера и метаэвристические алгоритмы.

1.1. ПОСТАНОВКА ЗАДАЧИ ИНТЕРВАЛЬНОЙ ε-МИНИМИЗАЦИИ

Классическая постановка задачи минимизации выглядит следующим образом: пусть имеется некоторый брус $\boldsymbol{s}$ и заданная функция $f:\mathbb{R}^n \to \mathbb{R}$; требуется найти точку x^*, такую что

$$f(x^*) = \min_{x \in \boldsymbol{s}} f(x) \Leftrightarrow \forall x \in \boldsymbol{s}: f(x^*) \le f(x). \qquad (1.1)$$

где брус – вектор, компоненты которого состоят из интервалов; условимся в дальнейшем для обозначения интервального вектора (бруса) использовать буквы латинского и греческого алфавита с полужирным начертанием: $\boldsymbol{x} = \boldsymbol{x}_1 \times \ldots \times \boldsymbol{x}_n = [\underline{\boldsymbol{x}}; \overline{\boldsymbol{x}}] = [\underline{\boldsymbol{x}_1}; \overline{\boldsymbol{x}_1}] \times \ldots \times [\underline{\boldsymbol{x}_n}; \overline{\boldsymbol{x}_n}] = \{x \in \mathbb{R}^n \mid \underline{\boldsymbol{x}} \le x \le \overline{\boldsymbol{x}}\}$. Множество брусов размерности n будем обозначать как $\mathbb{IR}^n$.

Для произвольного бруса $\boldsymbol{x}$ определены следующие характеристики: нижняя граница $\inf \boldsymbol{x} = \underline{\boldsymbol{x}} = (\underline{\boldsymbol{x}_1}, \ldots, \underline{\boldsymbol{x}_n})^{\mathrm{T}}$, верхняя граница $\sup \boldsymbol{x} = \overline{\boldsymbol{x}} = (\overline{\boldsymbol{x}_1}, \ldots, \overline{\boldsymbol{x}_n})^{\mathrm{T}}$, ширина $\mathrm{wid}\,\boldsymbol{x} = \mathrm{wid}(\boldsymbol{x}) = \overline{\boldsymbol{x}} - \underline{\boldsymbol{x}} = (\overline{\boldsymbol{x}_1} - \underline{\boldsymbol{x}_1}, \ldots, \overline{\boldsymbol{x}_n} - \underline{\boldsymbol{x}_n}) \ge 0$, радиус $\mathrm{rad}\,\boldsymbol{x} = \mathrm{rad}(\boldsymbol{x}) = \frac{1}{2} \cdot (\overline{\boldsymbol{x}} - \underline{\boldsymbol{x}}) = \frac{1}{2} \cdot (\overline{\boldsymbol{x}_1} - \underline{\boldsymbol{x}_1}, \ldots, \overline{\boldsymbol{x}_n} - \underline{\boldsymbol{x}_n})^{\mathrm{T}}$, средняя точка $\mathrm{mid}\,\boldsymbol{x} = \mathrm{mid}(\boldsymbol{x}) = \frac{1}{2} \cdot (\overline{\boldsymbol{x}} + \underline{\boldsymbol{x}}) = \frac{1}{2} \cdot (\overline{\boldsymbol{x}_1} + \underline{\boldsymbol{x}_1}, \ldots, \overline{\boldsymbol{x}_n} + \underline{\boldsymbol{x}_n})$. Кроме этого, добавлена следующая характеристика бруса (омега-характеристику): $\omega(\boldsymbol{x}) = \max_{i=1,\ldots,n} \mathrm{wid}(\boldsymbol{x}) = \max_{i=1,\ldots,n} (\overline{\boldsymbol{x}_1} - \underline{\boldsymbol{x}_1}, \ldots, \overline{\boldsymbol{x}_n} - \underline{\boldsymbol{x}_n})$.

Стоит отметить, что обычному вещественному числу $x = (x_1, \ldots, x_n)^{\mathrm{T}} \in \mathbb{R}^n$ соответствует брус $\boldsymbol{x} = [x; x] = [\boldsymbol{x}_1; \boldsymbol{x}_1] \times \ldots \times [\boldsymbol{x}_n; \boldsymbol{x}_n] \in \mathbb{IR}^n$.

Поскольку вместо независимых переменных используются интервалы, то вычисление функции производится уже средствами интервального анализа: находится интервальное расширение функции. Поэтому предлагается новая математическая модель задачи оптимизации – задача интервальной ε-минимизации.

Постановка задачи **интервальной** ε-минимизации выглядит следующим образом: пусть имеется некоторый брус $\boldsymbol{s}$ и заданная функция $f:\mathbb{R}^n \to \mathbb{R}$; требуется найти интервальный вектор $\boldsymbol{p}^*$, такой что

$$\boldsymbol{p}^* \subseteq \boldsymbol{s}, \omega(\boldsymbol{p}^*) \le \varepsilon, \nexists \boldsymbol{x} \subseteq \boldsymbol{s}, \omega(\boldsymbol{x}) \ge \varepsilon : \overline{\boldsymbol{f}(\boldsymbol{x})} < \underline{\boldsymbol{f}(\boldsymbol{p}^*)}, \tag{1.2}$$

где функция $\boldsymbol{f}:\mathbb{IR}^n \to \mathbb{IR}$ называется интервальным расширением функции $f:\mathbb{R}^n \to \mathbb{R}$ (это означает, что $\{f(\xi) \mid \xi \in \boldsymbol{x}\} \subseteq \boldsymbol{f}(\boldsymbol{x}), \forall \boldsymbol{x} \in \mathbb{IR}^n$). Интервальное расширение функции позволяет получить априорную оценку множества значений искомой функции. Если вместо переменных используются интервалы, а вместо арифметических операций и элементарных функций – их интервальные аналоги и расширения, то полученное расширение называется естественным.

Задачу (1.2) можно сформулировать следующим образом: требуется найти брус $\boldsymbol{p}^*$, омега-характеристика которого не превышает заданной величины ε, но при этом не должно существовать другого бруса, на котором верхняя граница значения естественного интервального расширения минимизируемой функции меньше, чем нижняя оценка этого расширения на брусе $\boldsymbol{p}^*$.

Известно, что при использовании интервального анализа возникает эффект зависимости, влияние которого можно исключить или уменьшить различными способами. Поскольку разрабатываемые в настоящей работе алгоритмы интервальной оптимизации применяются далее для решения задач синтеза оптимального управления динамическими системами, то возникает требование, чтобы результирующие брусы имели достаточно малую ширину, и их можно было использовать на практике. Это важно, так как значение координат вектора управления выбирается произвольно в полученном интервале. Следовательно, предполагается, что размеры брусов, участвующих в расчетах, достаточно малы, и поэтому эффект зависимости сказывается на результатах не значительно.

К достоинствам известных интервальных методов оптимизации следует отнести наличие гарантий того, что решение принадлежит полученному в результате брусу. В данной работе применяются два подхода. Первый опирается на использование инвертера и приводит к тому, что гарантируется принадлежность наилучшего значения целевой функции итоговому интервалу, однако принадлежность решения итоговому брусу не гарантируется. Второй подход реализуется с помощью эвристических процедур, управляемых с помощью правил, реализуемых на более высоком уровне иерархии, т.е. метаэвристических методов. При этом решение ищется в виде последовательности брусов. Принадлежность решения

полученному в результате наилучшему брусу не гарантируется. В то же время предложенные эвристические процедуры и механизм управления ими направлены на обеспечение выполнения условия (1.2), которое слабее требования принадлежности точки экстремума итоговому брусу. Введение условия (1.2) позволяет разрабатывать численные методы поиска экстремума, в которых очередное приближение координаты решения ищется в виде интервала неопределенности, что естественно с точки зрения вычислительной математики.

Замечания.

1. Если ε равно нулю, задача интервальной ε-минимизации (1.2) совпадет с задачей (1.1). Действительно, при $\varepsilon = 0$ брус $\boldsymbol{p}^*$ станет интервальным вектором нулевой ширины, т.е. точкой. Рассмотрим все брусы $\boldsymbol{x} \subseteq \boldsymbol{s}$ нулевой ширины. В этом случае эффект зависимости [97] пропадет и из (1.2) получаем $\forall \boldsymbol{x} \subseteq \boldsymbol{s} : \underline{\boldsymbol{f}(\boldsymbol{p}^*)} = \overline{\boldsymbol{f}(\boldsymbol{p}^*)} \le \underline{\boldsymbol{f}(\boldsymbol{x})} = \overline{\boldsymbol{f}(\boldsymbol{x})}$, что совпадает с (1.1).

2. Если множество допустимых решений в задаче минимизации задается ограничениями $e_i(x) = 0, g_j(x) \le 0, i = 1,...,c_e,\ j = 1,...,c_g$, то искомую функцию f предлагается заменить вспомогательной функцией F с использованием интервальных штрафных функций и ее соответствующим интервальным расширением:

$$\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}) + \sum_{i=1}^{c_e} R_i^e \cdot h_\infty^0(\boldsymbol{e}_i(\boldsymbol{x}),[0;0]) + \sum_{j=1}^{c_g} R_j^g \cdot h_\infty^0(\boldsymbol{g}_j(\boldsymbol{x}),]-\infty;0]), \qquad (1.3)$$

где $R_i^e, R_j^g, i = 1,\ldots,c_e,\ j = 1,\ldots c_g$ – параметры штрафов, а $h_\infty^0(\boldsymbol{a},\boldsymbol{b}) = \inf\{r \in \mathbb{R}^+ \mid \boldsymbol{a} \subset \boldsymbol{b} + r\cdot[-1;1]\}$ – мера близости двух интервалов. Второе слагаемое в правой части (1.3) характеризует степень выполнения ограничений-равенств, а третье – ограничений-неравенств.

1.2. ИНВЕРСНЫЕ МЕТОДЫ РЕШЕНИЯ ЗАДАЧИ ИНТЕРВАЛЬНОЙ ε-МИНИМИЗАЦИИ

В данном разделе разработаны интервальные алгоритмы оптимизации, использующие *инвертер* и названные соответственно инверсными. Инвертер $INV(f, \boldsymbol{s}, \boldsymbol{l}, \varepsilon)$ - функция, которая по заданному интервалу $\boldsymbol{l}$, функции f и параметру точности ε находит в области поиска $\boldsymbol{s}$ множество брусов $\mathcal{P}$ такое, что $\forall \boldsymbol{x} \in \mathcal{P}$ справедливо выражение $\boldsymbol{x} \subseteq \boldsymbol{s}, \omega(\boldsymbol{x}) \ge \varepsilon, \boldsymbol{f}(\boldsymbol{x}) \subseteq \boldsymbol{l}$ или выражение $\boldsymbol{x} \subseteq \boldsymbol{s}, \omega(\boldsymbol{x}) < \varepsilon, \boldsymbol{f}(\boldsymbol{x}) \cap \boldsymbol{l} \ne \varnothing$ [97].

Помимо инвертера инверсные алгоритмы в ходе работы часто оперируют процедурой *бисекции* бруса $\boldsymbol{p}$, в ходе которой образуются два новых интервальных вектора

$\boldsymbol{p}^1 = \boldsymbol{p}_1 \times \ldots \times [\underline{\boldsymbol{p}_k}; \mathrm{mid}(\boldsymbol{p}_k)] \times \ldots \times \boldsymbol{p}_n$ и $\boldsymbol{p}^2 = \boldsymbol{p}_1 \times \ldots \times [\mathrm{mid}(\boldsymbol{p}_k); \overline{\boldsymbol{p}_k}] \times \ldots \times \boldsymbol{p}_n$, где k – наименьший номер компоненты бруса $\boldsymbol{p}$ с наибольшей шириной.

1.2.1. МЕТОД ДИХОТОМИИ ЦЕЛЕВОГО ИНТЕРВАЛА

Стратегия поиска

Сначала считается значение естественного интервального расширения минимизируемой функции на области поиска (полученное значение рассматривается как целевой интервал). На каждой итерации алгоритма целевой интервал делится на две части. Далее применяется инвертер к первой части. Если выработанное им множество непустое, то проверяется условие точности (омега-характеристика целевого интервала должна быть меньше некоторого заданного значения). В случае его невыполнения первая часть принимается за новый целевой интервал, начинается новая итерация алгоритма. Если условие точности выполняется, то в выработанном инвертером множестве содержится брус, который содержит точку минимума. Если выработанное инвертером множество при применении к первой части пустое, то вторая часть принимается за новый целевой интервал, и алгоритм начинает новую итерацию.

Алгоритм

Шаг 1. Задать $\boldsymbol{s}$ – область поиска, ε – параметр точности, отвечающий за размер бруса во множестве $\mathcal{P}$, которое будет вырабатываться инвертером, и ζ – параметр точности остановки алгоритма.

Шаг 2. С помощью естественного интервального расширения $\boldsymbol{f}$ найти значение целевого интервала: $\boldsymbol{y} = f(\boldsymbol{s})$ на области поиска $\boldsymbol{s}$.

Шаг 3. Дихотомически разделить интервал $\boldsymbol{y}$ на два интервала $\boldsymbol{y}^1 = [\underline{\boldsymbol{y}}; \frac{\underline{\boldsymbol{y}} + \overline{\boldsymbol{y}}}{2}]$ и $\boldsymbol{y}^2 = [\frac{\underline{\boldsymbol{y}} + \overline{\boldsymbol{y}}}{2}; \overline{\boldsymbol{y}}]$.

Шаг 4. Инвертировать интервал $\boldsymbol{y}^1$. В ходе данного шага вырабатывается множество брусов $\mathcal{P} = INV(f, \boldsymbol{s}, \boldsymbol{y}^1, \varepsilon)$.

Шаг 5. Если множество $\mathcal{P}$, выработанное инвертером, непустое, то перейти к шагу 6, если пустое – к шагу 7.

Шаг 6. Если $\omega(\boldsymbol{y}^1) < \zeta$, тогда перейти к шагу 8. В противном случае положить интервал $\boldsymbol{y}$ равным $\boldsymbol{y}^1$ и вернуться к шагу 3.

Шаг 7. Положить интервал $\boldsymbol{y}$ равным $\boldsymbol{y}^2$ и если $\omega(\boldsymbol{y}) \geq \zeta$, то вернуться к шагу 3. В противном случае пересчитать $\mathcal{P} = INV(f, \boldsymbol{s}, \boldsymbol{y}, \varepsilon)$ и перейти к шагу 8.

Шаг 8. Выбор бруса.

Шаг 8.1. Положить $i = 1$, множество $\mathcal{P}_\varepsilon = \mathcal{P}$.

Шаг 8.2. Если $\omega(\boldsymbol{p}^i) \leq \varepsilon$, где $\boldsymbol{p}^i \in \mathcal{P}_\varepsilon$, то перейти к шагу 8.4.

Шаг 8.3. Добавить в множество $\mathcal{P}_\varepsilon$ брусы, полученные с помощью процедуры бисекции интервального вектора $\boldsymbol{p}^i$, удалить $\boldsymbol{p}^i$.

Шаг 8.4. Увеличить i на единицу. Если $i \leq |\mathcal{P}_\varepsilon|$, где $|\cdot|$ - мощность множества, то перейти к шагу 8.2.

Шаг 8.5. Для каждого $\boldsymbol{p}^i \in \mathcal{P}_\varepsilon$ найти $\boldsymbol{f}(\boldsymbol{p}^i)$ и выбрать $\boldsymbol{p}^* = \operatorname{Arg}\min\limits_{\boldsymbol{p}^i \in \mathcal{P}_\varepsilon} \underline{\boldsymbol{f}(\boldsymbol{p}^i)}$.

Рекомендации по подбору параметров

Параметры ε, ζ имеют одинаковое влияние на работу алгоритма. Уменьшение этих параметров приводит к одновременному увеличению точности и времени работы алгоритма. Увеличение же приводит к общему ускорению за счет снижения точности работы алгоритма.

1.2.2. МЕТОД ОТСЕЧКИ ВИРТУАЛЬНЫХ ЗНАЧЕНИЙ

Стратегия поиска

Естественное интервальное расширение функции может давать плохую оценку образа функции, и полученный интервал помимо истинных значений, которые принимает функция, может содержать значения, которые в принципе не могут быть достигнуты (эти значения будем называть виртуальными). Поэтому в ходе поиска требуется по возможности уменьшить множество виртуальных значений (отсечь их от полученной оценки образа).

Стратегия схожа со стратегией метода дихотомии целевого интервала. Единственное отличие – наличие стадии сжатия значения естественного интервального расширения минимизируемой функции (шаг 2), в ходе которой происходит разбиение искомой области поиска.

Алгоритм

Шаг 1. Задать $\boldsymbol{s}$ – область поиска, $\delta \in \{0\} \cup \mathbb{N}$ - параметр сетки, ε – параметр точности, отвечающий за размер бруса во множестве $\mathcal{P}$, которое будет вырабатываться инвертером, и ζ – параметр точности остановки алгоритма.

Шаг 2. *Оператор сжатия*. Вследствие наличия зависимости множества виртуальных значений от размера области поиска и структуры функции строится разбиение области поиска для последующего сжатия множества виртуальных значений следующим образом:

Шаг 2.1. Создать множество брусов $\mathcal{O} = \{\boldsymbol{s}\}$.

Шаг 2.2. Если δ равно нулю, то закончить построение разбиения и перейти к шагу 3. В противном случае – к шагу 2.3.

Шаг 2.3. Создать временные множества брусов $\mathcal{T}^1 = \mathcal{O}, \mathcal{T}^2 = \varnothing$.

Шаг 2.4. Положить $i = 1$.

Шаг 2.5. Каждый брус $\boldsymbol{t}^j \in \mathcal{T}^1, j = \overline{1, |\mathcal{T}^1|}$ ($|\cdot|$ - мощность множества) подвергнуть бисекции, результат добавить в $\mathcal{T}^2$.

Шаг 2.6. Положить $\mathcal{T}^1 = \mathcal{T}^2$. Увеличить i на единицу. Если $i > n$, то перейти к шагу 2.7, в противном случае – к шагу 2.5.

Шаг 2.7. Положить $\mathcal{O} = \mathcal{T}^1$. Уменьшить δ на единицу. Если δ равно нулю, то перейти к шагу 3, в противном случае – к шагу 2.3.

Шаг 3. Найти целевой интервал $\boldsymbol{y} = [\min\limits_{\boldsymbol{o} \in \mathcal{O}} \underline{\boldsymbol{f}(\boldsymbol{o})}, \max\limits_{\boldsymbol{o} \in \mathcal{O}} \overline{\boldsymbol{f}(\boldsymbol{o})}]$.

Шаг 4. Дихотомически разделить интервал $\boldsymbol{y}$ на два интервала $\boldsymbol{y}^1 = [\underline{\boldsymbol{y}}; \frac{\underline{\boldsymbol{y}} + \overline{\boldsymbol{y}}}{2}]$ и $\boldsymbol{y}^2 = [\frac{\underline{\boldsymbol{y}} + \overline{\boldsymbol{y}}}{2}; \overline{\boldsymbol{y}}]$.

Шаг 5. Инвертировать интервал $\boldsymbol{y}^1$. В ходе данного шага вырабатывается множество брусов $\mathcal{P} = INV(f, \boldsymbol{s}, \boldsymbol{y}^1, \varepsilon)$.

Шаг 6. Если множество $\mathcal{P}$, выработанное инвертером, то непустое, перейти к шагу 7, если пустое – к шагу 8.

Шаг 7. Если $\omega(\boldsymbol{y}^1) < \zeta$, тогда перейти к шагу 9. В противном случае положить интервал $\boldsymbol{y}$ равным $\boldsymbol{y}^1$ и вернуться к шагу 4.

Шаг 8. Положить интервал $\boldsymbol{y}$ равным $\boldsymbol{y}^2$ и если $\omega(\boldsymbol{y}) \geq \zeta$, то вернуться к шагу 4. В противном случае пересчитать $\mathcal{P} = INV(f, \boldsymbol{s}, \boldsymbol{y}, \varepsilon)$ и перейти к шагу 9.

Шаг 9. Выбор бруса.

Шаг 9.1. Положить $i = 1$, множество $\mathcal{P}_\varepsilon = \mathcal{P}$.

Шаг 9.2. Если $\omega(\boldsymbol{p}^i) \leq \varepsilon$, где $\boldsymbol{p}^i \in \mathcal{P}_\varepsilon$, то перейти к шагу 9.4.

Шаг 9.3. Добавить в множество $\mathcal{P}_\varepsilon$ брусы, полученные с помощью процедуры бисекции интервального вектора $\boldsymbol{p}^i$, удалить $\boldsymbol{p}^i$.

Шаг 9.4. Увеличить i на единицу. Если $i \leq |\mathcal{P}_\varepsilon|$, где $|\cdot|$ - мощность множества, то перейти к шагу 9.2.

Шаг 9.5. Для каждого $\boldsymbol{p}^i \in \mathcal{P}_\varepsilon$ найти $\boldsymbol{f}(\boldsymbol{p}^i)$ и выбрать $\boldsymbol{p}^* = \operatorname{Arg}\min_{\boldsymbol{p}^i \in \mathcal{P}_\varepsilon} \underline{\boldsymbol{f}(\boldsymbol{p}^i)}$.

Рекомендации по подбору параметров

Параметры ε, ζ имеют одинаковое влияние на работу алгоритма. Уменьшение этих параметров приводит к одновременному увеличению точности и времени работы алгоритма. Увеличение же приводит к общему ускорению за счет снижения точности работы алгоритма.

Параметр δ может уменьшить время работы алгоритма за счет удаления части виртуальных значений. Тем не менее, слишком большие значения приведут к общему замедлению работы, а также могут вызвать проблемы, связанные с нехваткой памяти компьютера, на котором запускается алгоритм.

1.2.3. МЕТОД СТОХАСТИЧЕСКОЙ ОТСЕЧКИ ВИРТУАЛЬНЫХ ЗНАЧЕНИЙ

Стратегия поиска

Стратегия метода аналогична стратегии метода отсечки виртуальных значений, отличие заключается в способе работы оператора сжатия. Метод использует стохастических подход, применение которого обосновано тем, что, ввиду отсутствия предварительного анализа функции, делается допущение, что границы множества виртуальных значений расположены случайно.

Алгоритм

Шаг 1. Задать $\boldsymbol{s}$ – область поиска, $\delta \in \{0\} \cup \mathbb{N}$ - количество циклов, ε – параметр точности, отвечающий за размер бруса во множестве $\mathcal{P}$, которое будет вырабатываться инвертером, и ζ – параметр точности остановки алгоритма.

Шаг 2. Положить $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{s})$, $p_l = p_r = \operatorname{mid}(\boldsymbol{y})$.

Шаг 3. *Оператор сжатия*. Вследствие наличия зависимости множества виртуальных значений от размера области поиска и структуры функции строится разбиение области поиска для последующего сжатия множества виртуальных значений следующим образом:

Шаг 3.1. Если δ равно нулю, то перейти к шагу 4, в противном случае – к шагу 3.2.

Шаг 3.2. Случайным образом взять две точки $\xi \sim U(\underline{\boldsymbol{y}}, p_l)$, $\xi_2 \sim U(p_r, \overline{\boldsymbol{y}})$, где $U(\cdot,\cdot)$ обозначает равномерное распределение. Выработать с помощью инвертера множества $\mathcal{P}_1 = INV(f, \boldsymbol{s}, [\underline{\boldsymbol{y}}; \xi_1], \varepsilon)$ и $\mathcal{P}_2 = INV(f, \boldsymbol{s}, [\xi_2; \overline{\boldsymbol{y}}], \varepsilon)$.

Шаг 3.3. Если множество $\mathcal{P}_1$ пустое, то положить $\boldsymbol{y} = [\xi_1; \overline{\boldsymbol{y}}]$, в противном случае положить $p_l = \xi_1$.

Шаг 3.4. Если множество $\mathcal{P}_2$ пустое, то положить $\boldsymbol{y} = [\underline{\boldsymbol{y}}; \xi_2]$, в противном случае положить $p_r = \xi_2$.

Шаг 3.5. Уменьшить δ на единицу. Перейти к шагу 3.1.

Шаг 4. Дихотомически разделить интервал $\boldsymbol{y}$ на два интервала $\boldsymbol{y}^1 = [\underline{\boldsymbol{y}}; \frac{\underline{\boldsymbol{y}} + \overline{\boldsymbol{y}}}{2}]$ и $\boldsymbol{y}^2 = [\frac{\underline{\boldsymbol{y}} + \overline{\boldsymbol{y}}}{2}; \overline{\boldsymbol{y}}]$.

Шаг 5. Инвертировать интервал $\boldsymbol{y}^1$. В ходе данного шага вырабатывается множество брусов $\mathcal{P} = INV(f, \boldsymbol{s}, \boldsymbol{y}^1, \varepsilon)$.

Шаг 6. Если множество $\mathcal{P}$, выработанное инвертером, непустое, то перейти к шагу 7, если пустое – к шагу 8.

Шаг 7. Если $\omega(\boldsymbol{y}^1) < \zeta$, тогда перейти к шагу 9. В противном случае положить интервал $[y]$ равным $\boldsymbol{y}^1$ и вернуться к шагу 4.

Шаг 8. Положить интервал $\boldsymbol{y}$ равным $\boldsymbol{y}^2$ и если $\omega(\boldsymbol{y}) \geq \zeta$, то вернуться к шагу 4. В противном случае пересчитать $\mathcal{P} = INV(f, \boldsymbol{s}, \boldsymbol{y}, \varepsilon)$ и перейти к шагу 9.

Шаг 9. Выбор бруса.

Шаг 9.1. Положить $i = 1$, множество $\mathcal{P}_\varepsilon = \mathcal{P}$.

Шаг 9.2. Если $\omega(\boldsymbol{p}^i) \leq \varepsilon$, где $\boldsymbol{p}^i \in \mathcal{P}_\varepsilon$, то перейти к шагу 9.4.

Шаг 9.3. Добавить в множество $\mathcal{P}_\varepsilon$ брусы, полученные с помощью процедуры бисекции интервального вектора $\boldsymbol{p}^i$, удалить $\boldsymbol{p}^i$.

Шаг 9.4. Увеличить i на единицу. Если $i \leq |\mathcal{P}_\varepsilon|$, где $|\cdot|$ - мощность множества, то перейти к шагу 9.2.

Шаг 9.5. Для каждого $\boldsymbol{p}^i \in \mathcal{P}_\varepsilon$ найти $\boldsymbol{f}(\boldsymbol{p}^i)$ и выбрать $\boldsymbol{p}^* = \operatorname{Arg}\min_{\boldsymbol{p}^i \in \mathcal{P}_\varepsilon} \underline{\boldsymbol{f}(\boldsymbol{p}^i)}$.

Рекомендации по подбору параметров

Параметры ε, ζ имеют одинаковое влияние на работу алгоритма. Уменьшение этих параметров приводит к одновременному увеличению точности и времени работы алгоритма. Увеличение же приводит к общему ускорению за счет снижения точности работы алгоритма.

Параметр δ может уменьшить время работы алгоритма за счет удаления части виртуальных значений. Тем не менее, слишком большие значения приведут к общему замедлению работы, а также могут вызвать проблемы, связанные с нехваткой памяти компьютера, на котором запускается алгоритм.

1.2.4. МЕТОД СТОХАСТИЧЕСКИХ ВЫРЫВАНИЙ

Стратегия поиска

Стратегия метода заключается в отбраковывании брусов в области поиска за счет «вырывания» из области поиска небольших интервальных векторов (омега-характеристика которых не превосходит некоторого, заранее объявленного значения). Для отбраковывания брусов используется инвертер. В процессе данного «вырывания» будет постепенно уменьшаться ширина целевого интервала, в котором хранится значение минимума функции. При достижении необходимой ширины целевой интервал инвертируется и из полученного множества выбирается решение.

Алгоритм

Шаг 1. Задать $\boldsymbol{s}$ – область поиска, ε – параметр точности (отвечает за размер выходного бруса), ζ – параметр точности остановки алгоритма, максимальное количество попыток A и максимальное количество проходов без улучшений W.

Шаг 2. Создать множество областей поиска $\mathcal{S} = \{\boldsymbol{s}\}$, положить величину $w = 0$ и найти целевой интервал $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{s})$.

Шаг 3. Если $\omega(\boldsymbol{y}) \geq \zeta$, то перейти к шагу 4. В противном случае сгенерировать множество $\mathcal{S}^* = \bigcup_{\boldsymbol{s}^i \in \mathcal{S}} INV(f, \boldsymbol{s}^i, \boldsymbol{y}, \varepsilon)$ и перейти к шагу 19.

Шаг 4. Сгенерировать A случайных номеров брусов $\{id^i\}_{i=1}^{A}$ и A соответствующих случайных точек $\{m^{i,id^i} = (m_1^{i,id^i}, \ldots, m_n^{i,id^i})^T \in \boldsymbol{s}^{id^i}\}_{i=1}^{A}$ (номер id^i – указывает на номер бруса $\boldsymbol{s}^{id^i}$ из $\mathcal{S}$, в котором генерировалась точка).

Шаг 5. На каждой точке m^{i,id^i} построить интервальный вектор $\boldsymbol{q}^i = ([m_1^{i,id^i} - \varepsilon/2; m_1^{i,id^i} + \varepsilon/2] \times \ldots \times [m_n^{i,id^i} - \varepsilon/2; m_n^{i,id^i} + \varepsilon/2]) \cap \boldsymbol{s}^{id^i}$, $i = 1, \ldots, A$.

Шаг 6. Из множества брусов $\{\boldsymbol{q}^i\}_{i=1}^{A}$ выбрать брус $\boldsymbol{q}^* = \operatorname{Arg}\min_{i=1,\ldots,A} \underline{\boldsymbol{f}(\boldsymbol{q}^i)}$.

Шаг 7. Если $\overline{\boldsymbol{f}(\boldsymbol{q}^*)} < \overline{\boldsymbol{y}}$, то перейти к шагу 8. В противном случае – к шагу 12.

Шаг 8. Положить $\boldsymbol{y} = [\underline{\boldsymbol{y}}; \overline{\boldsymbol{f}(\boldsymbol{q}^*)}]$.

Шаг 9. Создать множество $\mathcal{S}_t = \varnothing$.

Шаг 10. Для каждого из брусов $\boldsymbol{s}^i$ в $\mathcal{S}$ результат инвертирования $INV(f, \boldsymbol{s}^i, \boldsymbol{y}, \varepsilon)$ занести в множество $\mathcal{S}_t$.

Шаг 11. Заменить множество $\mathcal{S}$ множеством $\mathcal{S}_t$, положить $w = 0$. Перейти к шагу 13.

Шаг 12. Увеличить w на единицу.

Шаг 13. Если $w > W$, то перейти к шагу 14. В противном случае – к шагу 3.

Шаг 14. Уменьшить w на единицу.

Шаг 15. Дихотомически разделить интервал $\boldsymbol{y}$ на два интервала $\boldsymbol{y}^1 = [\underline{\boldsymbol{y}}; \frac{\boldsymbol{y} + \overline{\boldsymbol{y}}}{2}]$ и $\boldsymbol{y}^2 = [\frac{\boldsymbol{y} + \overline{\boldsymbol{y}}}{2}; \overline{\boldsymbol{y}}]$.

Шаг 16. Создать множество $\mathcal{S}_t = \varnothing$.

Шаг 17. Для каждого из брусов $\boldsymbol{s}^i$ в $\mathcal{S}$ результат инвертирования $INV(f, \boldsymbol{s}^i, \boldsymbol{y}^1, \varepsilon)$ занести в множество $\mathcal{S}_t$.

Шаг 18. Если множество $\mathcal{S}_t$ пусто, то положить $\boldsymbol{y} = \boldsymbol{y}^2$. В противном случае положить $\boldsymbol{y} = \boldsymbol{y}^1$ и заменить множество $\mathcal{S}$ множеством $\mathcal{S}_t$. Перейти к шагу 3.

Шаг 19. Выбор бруса.

Шаг 19.1. Положить $i = 1$, множество $\mathcal{S}_\varepsilon^* = \mathcal{S}^*$.

Шаг 19.2. Если $\omega(\boldsymbol{s}^i) \le \varepsilon$, где $\boldsymbol{s}^i \in \mathcal{S}_\varepsilon^*$, то перейти к шагу 19.4.

Шаг 19.3. Добавить в множество $\mathcal{S}_\varepsilon^*$ брусы, полученные с помощью процедуры бисекции интервального вектора $\boldsymbol{s}^i$, удалить $\boldsymbol{s}^i$.

Шаг 19.4. Увеличить i на единицу. Если $i \le |\mathcal{S}_\varepsilon^*|$, где $|\cdot|$ - мощность множества, то перейти к шагу 19.2.

Шаг 19.5. Для каждого $\boldsymbol{s}^i \in \mathcal{S}_\varepsilon^*$ найти $\boldsymbol{f}(\boldsymbol{s}^i)$ и выбрать $\boldsymbol{s}^* = \operatorname{Arg}\min\limits_{\boldsymbol{s}^i \in \mathcal{S}_\varepsilon^*} \underline{\boldsymbol{f}(\boldsymbol{s}^i)}$.

Рекомендации по подбору параметров

Параметры ε, ζ, A, W имеют одинаковое влияние на работу алгоритма. Уменьшение этих параметров приводит к одновременному увеличению точности и времени работы

алгоритма. Увеличение же приводит к общему ускорению за счет снижения точности работы алгоритма.

1.2.5. ОБОБЩЕННЫЙ ИНВЕРСНЫЙ МЕТОД

Стратегия поиска

Этот метод обладает высокой гибкостью и настраиваемостью в связи с тем, что операторы сжатия и проверки вынесены в отдельные модули и могут выбираться в зависимости от условий задачи. Предложены следующие операторы:

- операторы проверки:
    - Original Inverter operator (OI – operator) – базовая проверка, использующаяся в методе дихотомии целевого интервала;
    - Original Inverter with Renewal operator (OIR – operator) – в дополнение к базовой проверке, выполняемой после каждой успешной операции, множество брусов $\mathcal{X}$, подаваемых на вход, заменяется на множество, выработанное инвертером;
    - First True operator (FT – operator) – проверка до первого подходящего элемента;
    - First True with Renewal operator (FTR – operator) – проверка до первого подходящего элемента, множество брусов $\mathcal{X}$, подаваемых на вход, заменяется на множество, выработанное инвертером;
    - Random Sample operator (RS – operator) – эвристический оператор, который использует стохастический подход вместо инвертера;
    - Random Sample with Renewal operator (RSR – operator) – эвристический оператор, который использует стохастический подход вместо инвертера и модифицирует множество брусов $\mathcal{X}$, подаваемых на вход;
- операторы сжатия:
    - Search Area Split operator (SAS - operator) – сжатие на основе разбиения области поиска;
    - Random Point Sample operator (RPS – operator) – сжатие на основе значений функций в случайно сгенерированных интервальных векторах нулевой ширины.
    - Inverter Cut operator (IC – operator) – сжатие на основе инвертера;
    - Inverter Cut with First True Check operator (ICFTC – operator) – сжатие на основе проверки до первого подходящего бруса;

- Inverter Cut with First True Check with Renewal operator (ICFTCR – operator) – сжатие на основе проверки до первого подходящего интервального вектора с обновлением множества брусов $\mathcal{X}$, попадающих на вход;

Следует упомянуть, что если использовать неэвристические операторы, то сходимость метода такая же, как и у метода дихотомии целевого интервала. Основным достоинством данного обобщенного метода является то, что его компоненты могут быть выбраны эвристическими, что позволить сократить общее время работы метода.

Алгоритмы операторов проверки

На вход всех операторов проверки подается множество брусов $\mathcal{X}$ и проверяемый интервал $\boldsymbol{l}$. Результатом работы является статус проверки: удачный свидетельствует о том, что в множестве брусов $\mathcal{X}$ есть брус $\boldsymbol{p}$, для которого выполнено $\omega(\boldsymbol{p}) \geq \varepsilon$ и $\boldsymbol{f}(\boldsymbol{p}) \subseteq \boldsymbol{l}$ или $\omega(\boldsymbol{p}) < \varepsilon$ и $\boldsymbol{f}(\boldsymbol{p}) \cap \boldsymbol{l} \neq \varnothing$.

Алгоритм OI оператора (для работы необходим параметр ширины w).

Шаг 1. Для каждого бруса из множества $\mathcal{X}$ применить алгоритм SIVIA [97] для интервала $\boldsymbol{l}$ с параметром w. Таким образом, будет выработано множество множеств брусов $\{\mathcal{P}_i\}_{i=1}^{|\mathcal{X}|}$ (где $|\cdot|$ - мощность множества) .

Шаг 2. Если $\bigcup_i \mathcal{P}_i \neq \varnothing$, то проверка выполнена удачно. В противном случае проверка завершена неудачно.

Алгоритм OIR оператора (для работы необходим параметр ширины w).

Шаг 1. Для каждого бруса $\boldsymbol{x}^i \in \mathcal{X}$ применить алгоритм SIVIA [97] для интервала $\boldsymbol{l}$ с параметром w. Результатом применения будут множества брусов $\{\mathcal{P}_i\}_{i=1}^{|\mathcal{X}|}$ (где $|\cdot|$ - мощность множества).

Шаг 2. Найти множество $\mathcal{X} = \bigcup_i \mathcal{P}_i$.

Шаг 3. Если $\mathcal{X} = \varnothing$, тогда проверка завершена неудачно. В противном случае проверка завершена удачно, заменить заданное множество $\mathcal{X}$ на найденное множество $\mathcal{X}$.

Алгоритм процедуры SIVIA_check (для работы необходим параметр ширины w).

Шаг 1. Для каждого бруса $\boldsymbol{x} \in \mathcal{X}$ найти $\boldsymbol{f}(\boldsymbol{x})$.

Шаг 2. Проверить выполнение следующих условий (сам брус $\boldsymbol{x}$ после проверки условий удаляется из $\mathcal{X}$):

1) если $\boldsymbol{f}(\boldsymbol{x}) \cap \boldsymbol{l} = \varnothing$, то продолжить работу;
2) если $\boldsymbol{f}(\boldsymbol{x}) \subseteq \boldsymbol{l}$ или $\boldsymbol{f}(\boldsymbol{x}) \cap \boldsymbol{l} \neq \varnothing, \omega(\boldsymbol{x}) < w$, то процедура завершена удачно;

3) в случае невыполнения условий 1 и 2 найти брусы $\boldsymbol{x}^1, \boldsymbol{x}^2$, полученные с помощью процедуры бисекции интервального вектора $\boldsymbol{x}$, и добавить их в множество $\mathcal{X}$.

Шаг 3. Если $\mathcal{X} = \varnothing$, то перейти к шагу 1, в противном случае – процедура завершена неудачно.

Алгоритм FT оператора (для работы необходим параметр ширины w).

Шаг 1. Положить $i = 1$.

Шаг 2. Задать множество $\mathcal{P} = \{\boldsymbol{p}^i \in \mathcal{X}\}$ применить процедуру SIVIA_check для интервала $\boldsymbol{l}$ с параметром w. Если процедура SIVIA_check завершилась удачно – проверка завершена удачно, в противном случае – к перейти к шагу 3.

Шаг 3. Если $i < |\mathcal{X}|$ (где $|\cdot|$ - мощность множества), то увеличить i на единицу и перейти к шагу 2, в противном случае – перейти к шагу 4.

Шаг 4. Генерировать сообщение о том, что проверка завершена неудачно.

Алгоритм FTR оператора (для работы необходим параметр ширины w).

Шаг 1. Задать множество $\mathcal{P} = \{\boldsymbol{p}^i \in \mathcal{X}\}$.

Шаг 2. Применить процедуру SIVIA_check для интервала $\boldsymbol{l}$ с параметром w для всех элементов из $\mathcal{X}$. Как только будет получена первая удачная проверка – закончить алгоритм, проверка завершена удачно. Иначе перейти к шагу 3.

Шаг 3. Заменить $\mathcal{X}$ на $\mathcal{P}$, проверка завершена неудачно.

Замечание. Обратная замена $\mathcal{X}$ на $\mathcal{P}$ необходима, так как при неудачном завершении проверки процедура SIVIA_check сделает множество $\mathcal{X}$ равным пустому множеству и с данным множеством невозможно будет продолжать работу. Замена на шаге 3 необходима, чтобы вернуть множество брусов $\mathcal{X}$ к своему исходному состоянию.

Алгоритм RS оператора (для работы необходимо задать количество попыток a, параметр ширины w).

Шаг 1. Задать $n = 1$.

Шаг 2. Если $n < a$, то перейти к шагу 3, в противном случае перейти к шагу 6.

Шаг 3. В случайно выбранном брусе $\boldsymbol{x} \in \mathcal{X}$ случайно сгенерировать брус $\boldsymbol{r} = [\xi_1; \zeta_1] \times \ldots \times [\xi_n; \zeta_n]$, где ξ_i, ζ_i – равномерно распределенные на интервале $\boldsymbol{x}_i$ независимые случайные величины (если $\xi_i > \zeta_i$, то i-й интервал заменяется на $[\zeta_i; \xi_i]$).

Шаг 4. Если $\boldsymbol{f}(\boldsymbol{r}) \subseteq \boldsymbol{l}$ или $\boldsymbol{f}(\boldsymbol{r}) \cap \boldsymbol{l} \neq \varnothing, \omega(\boldsymbol{r}) < w$, то закончить работу алгоритма, проверка завершена удачно.

Шаг 5. Увеличить n на единицу. Перейти к шагу 2.

Шаг 6. Проверка завершена неудачно.

Алгоритм RSR оператора (для работы необходимо задать количество попыток a, параметр ширины w).

Шаг 1. Задать $n=1$, $t=0$, множество $\mathcal{P}=\varnothing$.

Шаг 2. Если $n<a$, то перейти к шагу 3, в противном случае перейти к шагу 6.

Шаг 3. В случайно выбранном брусе $\boldsymbol{x}\in\mathcal{X}$ случайно сгенерировать интервальный вектор $\boldsymbol{r}=[\xi_1;\zeta_1]\times\ldots\times[\xi_n;\zeta_n]$, где ξ_i,ζ_i – равномерно распределенные на интервале $\boldsymbol{x}_i$ независимые случайные величины (если $\xi_i>\zeta_i$, то i-й интервал заменяется на $[\zeta_i;\xi_i]$).

Шаг 4. Если $\boldsymbol{f}(\boldsymbol{r})\subseteq\boldsymbol{l}$ или $\boldsymbol{f}(\boldsymbol{r})\cap\boldsymbol{l}\neq\varnothing,\omega(\boldsymbol{r})<w$, то положить $t=1$ и добавить $\boldsymbol{r}$ в $\mathcal{P}$.

Шаг 5. Увеличить n на единицу. Перейти к шагу 2.

Шаг 6. Если $t=1$, то проверка завершена удачно (заменить множество $\mathcal{X}$ на $\mathcal{P}$). В противном случае проверка завершена неудачно.

Алгоритмы операторов сжатия

На вход всех операторов сжатия подается брус $\boldsymbol{s}$. Результатом работы является интервал $\boldsymbol{y}$.

Алгоритм SAS оператора (для работы необходим параметр ширины w).

Шаг 1. Задать $\boldsymbol{y}=\varnothing$ - результирующий сжатый интервал.

Шаг 2. Для каждой из компонент вектора $\boldsymbol{s}_i$ найти наименьшее целое число n_i, такое что $\omega(\boldsymbol{s}_i)/n_i\le w$.

Шаг 3. Каждый из интервалов $\boldsymbol{s}_i$ представить в виде объединения непересекающихся интервалов $\bigcup_{j=1}^{n_i}\boldsymbol{s}_i^j$. Таким образом, создается разбиение изначального бруса $\boldsymbol{s}=\bigcup_{k=1}^{N}\boldsymbol{s}^k$, где $N=\prod_{i=1}^{n}n_i$.

Шаг 4. Найти $\boldsymbol{y}=\bigcup_{k=1}^{N}\boldsymbol{f}(\boldsymbol{s}^k)$.

Алгоритм RPS оператора (для работы необходимо число точек A).

Шаг 1. Найти $\boldsymbol{y}=\boldsymbol{f}(\boldsymbol{s})$ – результирующий сжатый интервал.

Шаг 2. Создать A случайных точек $\xi_i\in\boldsymbol{s},i=1,\ldots,A$.

Шаг 3. Найти $\boldsymbol{y}=[\underline{\boldsymbol{y}};\min_{i=1,\ldots,A}f(\xi_i)]$.

Алгоритм IC оператора (для работы необходимо задать количество точек A и параметр ширины w).

Шаг 1. Найти $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{s})$. Задать $a = 1$.

Шаг 2. Если $a < A$, сгенерировать случайную точку $\xi \in \boldsymbol{y}$. В противном случае завершить работу.

Шаг 3. Если алгоритм SIVIA (см. разд. П.2), примененный к интервалу $[\underline{\boldsymbol{y}}; \xi]$ с параметром w сгенерирует непустое множество, тогда $\boldsymbol{y} = [\underline{\boldsymbol{y}}; \xi]$. В противном случае $\boldsymbol{y} = [\xi; \overline{\boldsymbol{y}}]$.

Шаг 4. Увеличить a на единицу. Перейти к шагу 2.

Алгоритм ICFTC оператора (для работы необходимо задать количество точек A и параметр ширины w).

Шаг 1. Найти $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{s})$. Задать $a = 1$.

Шаг 2. Если $a < A$, сгенерировать случайную точку $\xi \in \boldsymbol{y}$. В противном случае завершить работу.

Шаг 3. Если процедура SIVIA_check, примененная к интервалу $[\underline{\boldsymbol{y}}; \xi]$ с параметром w завершится удачно, тогда $\boldsymbol{y} = [\underline{\boldsymbol{y}}; \xi]$. В противном случае $\boldsymbol{y} = [\xi; \overline{\boldsymbol{y}}]$.

Шаг 4. Увеличить a на единицу. Перейти к шагу 2.

Алгоритм ICFTCR оператора (для работы необходимо задать количество точек A и параметр ширины w).

Шаг 1. Найти $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{s})$. Задать $a = 1$.

Шаг 2. Если $a < A$, сгенерировать случайную точку $\xi \in \boldsymbol{y}$. В противном случае завершить работу.

Шаг 3. Задать множество $\mathcal{P} = \mathcal{X}$. Если процедура SIVIA_check, примененная к интервалу $[\underline{\boldsymbol{y}}; \xi]$ с параметром w и с множеством $\mathcal{X}$ завершится удачно, тогда $\boldsymbol{y} = [\underline{\boldsymbol{y}}; \xi]$. В противном случае $\boldsymbol{y} = [\xi; \overline{\boldsymbol{y}}]$, заменить $\mathcal{X}$ на $\mathcal{P}$.

Шаг 4. Увеличить a на единицу. Перейти к шагу 2.

Замечание. Обратная замена $\mathcal{X}$ на $\mathcal{P}$ необходима, так как при неудачном завершении проверки процедура SIVIA_check сделает множество $\mathcal{X}$ равным пустому множеству и с данным множеством невозможно будет продолжать работу. Замена на шаге 3 необходима, чтобы вернуть множество брусов $\mathcal{X}$ к своему исходному состоянию.

Алгоритм

Шаг 1. Задать $\boldsymbol{s}$ – область поиска, ε – параметр точности, отвечающий за размер брусов во множестве $\mathcal{P}$, которое будет вырабатываться инвертером; ζ – параметр точности остановки алгоритма. Выбрать операторы сжатия и проверки, задать для них параметры. Задать множество $\mathcal{P} = \{\boldsymbol{s}\}$.

Шаг 2. С помощью естественной интервального расширения минимизируемой функции f найти целевой интервал: $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{s})$.

Шаг 3. Заменить полученный интервал $\boldsymbol{y}$ интервалом $\boldsymbol{y}$, полученным с помощью выбранного оператора сжатия.

Шаг 4. Дихотомически разделить интервал $\boldsymbol{y}$ на два интервала $\boldsymbol{y}^1 = [\underline{\boldsymbol{y}}; \frac{\underline{\boldsymbol{y}} + \overline{\boldsymbol{y}}}{2}]$ и $\boldsymbol{y}^2 = [\frac{\underline{\boldsymbol{y}} + \overline{\boldsymbol{y}}}{2}; \overline{\boldsymbol{y}}]$.

Шаг 5. Проверить интервал $\boldsymbol{y}^1$ выбранным оператором проверки. Если проверка завершилась удачно, то перейти к шагу 6, в противном случае – к шагу 7.

Шаг 6. Если $\omega(\boldsymbol{y}^1) < \zeta$, тогда найти $\mathcal{P} = INV(f, \boldsymbol{s}, \boldsymbol{y}^1, \varepsilon)$ и перейти к шагу 8. В противном случае положить интервал $\boldsymbol{y}$ равным $\boldsymbol{y}^1$ и вернуться к шагу 4.

Шаг 7. Положить интервал $\boldsymbol{y}$ равным $\boldsymbol{y}^2$ и если $\omega(\boldsymbol{y}) \geq \zeta$, то вернуться к шагу 4. В противном случае пересчитать $\mathcal{P} = INV(f, \boldsymbol{s}, \boldsymbol{y}, \varepsilon)$ и перейти к шагу 8.

Шаг 8. Выбор бруса.

Шаг 8.1. Положить $i = 1$, множество $\mathcal{P}_\varepsilon = \mathcal{P}$.

Шаг 8.2. Если $\omega(\boldsymbol{p}^i) \leq \varepsilon$, где $\boldsymbol{p}^i \in \mathcal{P}_\varepsilon$, то перейти к шагу 8.4.

Шаг 8.3. Добавить в множество $\mathcal{P}_\varepsilon$ брусы, полученные с помощью процедуры бисекции интервального вектора $\boldsymbol{p}^i$, удалить $\boldsymbol{p}^i$.

Шаг 8.4. Увеличить i на единицу. Если $i \leq |\mathcal{P}_\varepsilon|$, где $|\cdot|$ - мощность множества, то перейти к шагу 8.2.

Шаг 8.5. Для каждого $\boldsymbol{p}^i \in \mathcal{P}_\varepsilon$ найти $\boldsymbol{f}(\boldsymbol{p}^i)$ и выбрать $\boldsymbol{p}^* = \operatorname{Arg}\min_{\boldsymbol{p}^i \in \mathcal{P}_\varepsilon} \underline{\boldsymbol{f}(\boldsymbol{p}^i)}$.

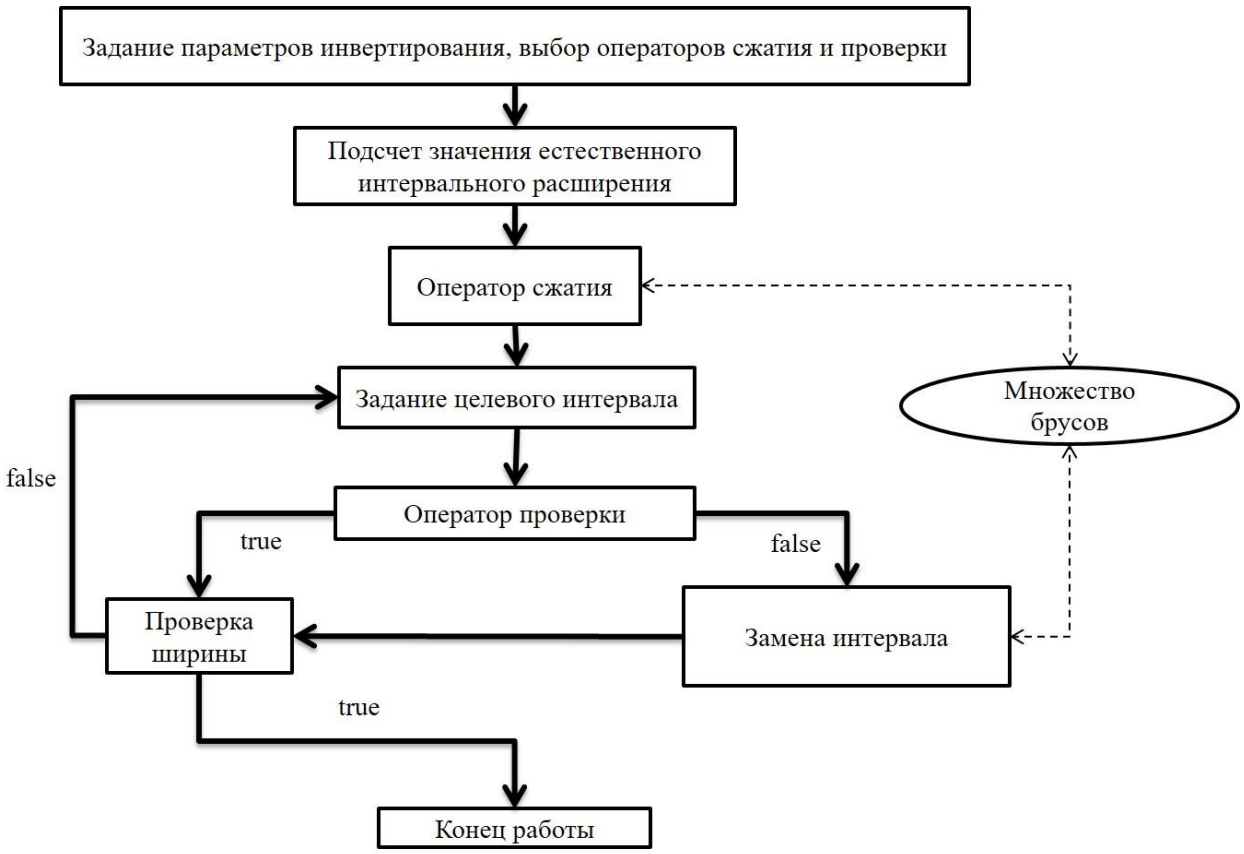


Рис. 1.1. Схема работы обобщенного инверсного метода

Рекомендации по подбору параметров

Параметры ε, ζ имеют одинаковое влияние на работу алгоритма. Уменьшение этих параметров приводит к одновременному увеличению точности и времени работы алгоритма. Увеличение же приводит к общему ускорению за счет снижения точности работы алгоритма.

При выборе операторов проверки и сжатия предпочтение следует отдать процедурам с обновлением, т.к. им не требуется постоянный пересчет уже проанализированных областей. Однако, такие процедуры требует больше памяти, что может негативно сказаться при недостатке этого ресурса.

Кроме того, использование эвристических процедур позволит ускорить работу алгоритма за счет сравнительно незначительного уменьшения точности работы.

1.2.6. ТЕОРЕМЫ О СВОЙСТВАХ РЕШЕНИЙ ИНТЕРВАЛЬНОЙ Ε-МИНИМИЗАЦИИ ИНВЕРСНЫМИ МЕТОДАМИ

Выделим свойства, характерные для всех инверсных алгоритмов:

- в ходе работы алгоритмов вырабатывается множество брусов $\mathcal{P}_\varepsilon = \{\boldsymbol{p}^i \mid \omega(\boldsymbol{p}^i) \le \varepsilon, \boldsymbol{f}(\boldsymbol{p}^i) \cap \boldsymbol{y} \ne \varnothing\}$, где $\boldsymbol{y}$ – последний целевой интервал, ε – параметр точности,

- генерируется последовательность из N вложенных целевых интервалов (конец процедуры генерации последовательности заканчивается при выполнении условия $\omega(\boldsymbol{y}) < \zeta$): интервал $\boldsymbol{y}$ дихотомически делится на два интервала $\boldsymbol{y}^1 = [\underline{\boldsymbol{y}}; \frac{\boldsymbol{y} + \overline{\boldsymbol{y}}}{2}]$ и $\boldsymbol{y}^2 = [\frac{\boldsymbol{y} + \overline{\boldsymbol{y}}}{2}; \overline{\boldsymbol{y}}]$, далее один из интервалов $\boldsymbol{y}^1$ или $\boldsymbol{y}^2$ рассматривается как целевой, и процедура повторяется над ним, и так далее (при этом новым целевым интервалом становится $\boldsymbol{y}^1$, если $INV(f, \boldsymbol{s}, \boldsymbol{y}^1, \varepsilon) \neq \varnothing$, а в противном случае – $\boldsymbol{y}^2$).

Таким образом, используя выделенные выше свойства, можно записать следующие теоремы.

Теорема 1. Пусть дана последовательность интервалов $\{\boldsymbol{y}^k\}_{k=1}^N$, таких что $\forall k = 1,..., N-1: \boldsymbol{y}^{k+1} \subset \boldsymbol{y}^k$, $\forall k = 1,..., N: \; INV(f, \boldsymbol{s}, \boldsymbol{y}^k, \varepsilon) \neq \varnothing$ (множество, вырабатываемое инвертером непустое), а при этом $INV(f, \boldsymbol{s}, [\underline{\boldsymbol{y}^k} - \omega(\boldsymbol{y}^k); \underline{\boldsymbol{y}^k}], \varepsilon) = \varnothing$ (множество, вырабатываемое инвертером пустое), и $\omega(\boldsymbol{y}^N) < \zeta$. Тогда $\boldsymbol{p}^* = \operatorname{Arg}\min\limits_{\boldsymbol{p} \in \mathcal{P}_\varepsilon} \underline{\boldsymbol{f}(\boldsymbol{p})}$ – решение задачи интервальной ε-минимизации, где $\mathcal{P}_\varepsilon = \{\boldsymbol{p}_\varepsilon \mid \omega(\boldsymbol{p}_\varepsilon) \le \varepsilon, \exists! i: \boldsymbol{p}_\varepsilon \subseteq \boldsymbol{p}^i \in \mathcal{P}\}$, $\mathcal{P} = INV(f, \boldsymbol{s}, \boldsymbol{y}^N, \varepsilon)$.

Доказательство теоремы 1. Вследствие того что в инверсных методах брус $\boldsymbol{p}^*$ выбирается из множества $\mathcal{P}_\varepsilon$, его ширина никогда не превысит выбранного значения ε (из-за способа построения множества $\mathcal{P}_\varepsilon$), т.е. $\omega(\boldsymbol{p}^*) \le \varepsilon$. По определению способа построения последовательности $\{\boldsymbol{y}^k\}_{k=1}^N$ для любого значения k не существует интервала $\tilde{\boldsymbol{y}} = [\underline{\boldsymbol{y}^k} - \omega(\boldsymbol{y}^k); \underline{\boldsymbol{y}^k}]$, такого что $INV(f, \boldsymbol{s}, \tilde{\boldsymbol{y}}, \varepsilon) \neq \varnothing$. Таким образом, по определению инвертера не существует $\boldsymbol{x} \subseteq \boldsymbol{s}$, для которого выполнено $\omega(\boldsymbol{x}) \ge \varepsilon, \boldsymbol{f}(\boldsymbol{x}) \subseteq \tilde{\boldsymbol{y}}$ или $\omega(\boldsymbol{x}) < \varepsilon, \boldsymbol{f}(\boldsymbol{x}) \cap \tilde{\boldsymbol{y}} \neq \varnothing$. Отсюда следует, что не существует $\boldsymbol{x} \subseteq \boldsymbol{s}$, для которого $\overline{\boldsymbol{f}(\boldsymbol{x})} \in \tilde{\boldsymbol{y}}$, что исключает возможность существования такого бруса $\boldsymbol{x} \subseteq \boldsymbol{s}, \omega(\boldsymbol{x}) \ge \varepsilon$, для которого выполнено условие $\overline{\boldsymbol{f}(\boldsymbol{x})} < \underline{\boldsymbol{f}(\boldsymbol{p}^*)}$.

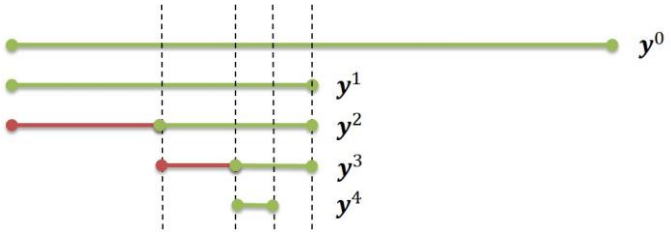


Рис. 1.2. Способ построения вложенных интервалов в инверсных алгоритмах (красным выделен интервал, если инвертер для него вырабатывает пустое множество)

Следствие из теоремы 1. Решение, полученное с помощью инверсных методов оптимизации, является решением задачи интервальной ε-минимизации. Действительно, процедура построения последовательности инвертируемых интервалов полностью удовлетворяет условиям теоремы 1 вследствие того, что она получена путем постоянного дихотомического деления пополам, и при этом выбирается исключительно тот интервал, для которого инвертер выработал непустое множество (рис. 1.2). Аналогично по способу построения множество $\mathcal{P}_\varepsilon$ удовлетворяет условиям теоремы, т.к. оно строится за счет рекурсивного применения процедуры бисекции.

Теорема 2. Пусть $x^* = \operatorname{Arg}\min_{x \in \boldsymbol{s}} f(x)$, а $\boldsymbol{p}^*$ - решение задачи интервальной ε-минимизации, удовлетворяющее условиям теоремы 1, тогда $f(x^*) \in \boldsymbol{f}(\boldsymbol{p}^*)$.

Доказательство теоремы 2. Для доказательства теоремы требуется показать, что для бруса $\boldsymbol{p}^*$ одновременно выполняются два неравенства: $f(x^*) \geq \underline{\boldsymbol{f}(\boldsymbol{p}^*)}$ и $f(x^*) \leq \overline{\boldsymbol{f}(\boldsymbol{p}^*)}$.

Докажем выполнение первого неравенства. Пусть $f(x^*) < \underline{\boldsymbol{f}(\boldsymbol{p}^*)}$, тогда существует $\tilde{\boldsymbol{p}}$ – решение задачи интервальной ε-минимизации, такой что $f(x^*) \in \boldsymbol{f}(\tilde{\boldsymbol{p}})$. Тогда получаем, что $\underline{[f](\tilde{\boldsymbol{p}})} < \underline{[f](\boldsymbol{p}^*)}$, что невозможно, так как $\boldsymbol{p}^*$ так же является решением задачи интервальной ε-минимизации и при этом $\boldsymbol{p}^* = \operatorname{Arg}\min_{\boldsymbol{p} \in \mathcal{P}_\varepsilon} \underline{[f](\boldsymbol{p})}$, таким образом первое неравенство выполнено.

Докажем выполнение второго неравенства. Пусть $f(x^*) > \overline{\boldsymbol{f}(\boldsymbol{p}^*)}$, тогда $\boldsymbol{f}(\boldsymbol{p}^*) = \varnothing$, что невозможно, так как $\boldsymbol{p}^* \neq \varnothing$, таким образом второе неравенство выполнено.

Так как оба неравенства выполнены, теорема доказана.

Следствие из теоремы 2. В брусе $\boldsymbol{p}^*$, полученном инверсными методами оптимизации, есть точка x, такая что $f(x) - f(x^*) \leq \omega(\boldsymbol{f}(\boldsymbol{p}^*))$.

1.3. МЕТАЭВРИСТИЧЕСКИЕ МЕТОДЫ РЕШЕНИЯ ЗАДАЧИ ИНТЕРВАЛЬНОЙ ε-МИНИМИЗАЦИИ

В данном разделе разработаны алгоритмы интервальной оптимизации, относящиеся к классу метаэвристических алгоритмов оптимизации [89]. Целью их создания является упрощение вычислительных процедур и, как следствие, сокращение вычислительных затрат без существенной потери точности.

1.3.1. МЕТОД УСРЕДНЕННЫХ КОНЦОВ ПУТЕЙ

Стратегия поиска

В основе метода усредненных концов путей лежат следующие процедуры:

1) построение случайной сетки (представляет собой разбиение бруса $\boldsymbol{s}$ на множество $\{\boldsymbol{n}^i\}_{i=1}^N$ непересекающихся брусов, таких что $\boldsymbol{s} = \bigcup_{i=1}^{N} \boldsymbol{n}^i$ на области поиска);
2) выбор случайного бруса из сетки;
3) поиск пути до «оптимального» решения (который рассматривается как последний брус) по сетке, начинающегося от выбранного бруса. Все последующие брусы должны иметь общую сторону с предыдущим интервальным вектором, а нижняя граница значения естественного интервального расширения минимизируемой функции уменьшаться.

Полученный в ходе выполнения описанной процедуры «оптимальный» брус запоминается. Процедура повторяется несколько раз. Из полученных брусов берутся их средние точки, на основе которых строится новый целевой брус (он рассматривается как наиболее вероятное, среднее положение завершения пути, начатого из любой области), к которому снова применяется описанная ранее процедура. Так повторяется до тех пор, пока омега-характеристика целевого бруса не будет удовлетворять условию точности.

Алгоритм

Шаг 1. Задать $\boldsymbol{s}$ – область поиска, ε – параметр точности (отвечает за размер выходного бруса), w – «скорость» уменьшения бруса, p_{amount} – количество точек для построения случайной сетки, $p_{attempts}$ – количество повторений, $r_{\max}$ – максимальное количество возвращений (используется во время поиска пути до «оптимального» решения). Положить целевой брус $\boldsymbol{t} = \boldsymbol{s}$.

Шаг 2. Если $\omega(\boldsymbol{t}) < \varepsilon$ – закончить работу алгоритма, вернув брус $\boldsymbol{t}$, в противном случае положить множество концов путей $\mathcal{E} = \varnothing$, перейти к шагу 3.

Шаг 3. Случайным образом, используя равномерное распределение, выбрать в целевом брусе p_{amount} точек.

Шаг 4. *Операция разбиения относительно выбранных случайных точек*. Разбить целевой брус относительно выбранных на шаге 3 точек на множество брусов (так как любой брус по сути является n-мерным параллелепипедом, то разбиение можно делать следующим образом: в каждой точке строятся n гиперплоскостей, каждая из которых параллельна одной

паре сторон данного параллелепипеда; совокупность этих гиперплоскостей и делит исходный брус).

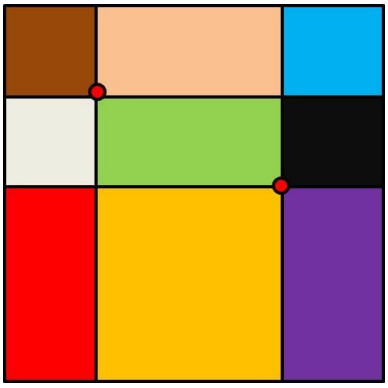


Рис. 1.3. Разбиение бруса случайными точками

Шаг 5. Из полученной сетки выбрать случайный брус $\boldsymbol{p}$.

Шаг 6. Процедура поиска пути от бруса $\boldsymbol{p}$ к «оптимальному» брусу:

Шаг 6.1. Создать множество брусов $\mathcal{N} = \{\boldsymbol{n}^i\}_{i=1}^{l}$, граничащих (имеющих общую сторону) с брусом $\boldsymbol{p}$. Задать список брусов, образующих путь, $\mathcal{P} = \{\boldsymbol{p}\}$; количество повторений $r = 0$.

Шаг 6.2. Если $r \le r_{\max}$, перейти к шагу 6.3, в противном случае перейти к шагу 6.6.

Шаг 6.3. Выбрать случайный брус $\boldsymbol{n}^j \in \mathcal{N}$, удалить его из $\mathcal{N}$. Если $\underline{\boldsymbol{f}(\boldsymbol{n}^j)} \le \underline{\boldsymbol{f}(\boldsymbol{p})}$, перейти к шагу 6.4, в противном случае перейти к шагу 6.5.

Шаг 6.4. Если $\boldsymbol{n}^j \in \mathcal{P}$, то положить $r = r + 1$, если $\boldsymbol{n}^j \notin \mathcal{P}$, то добавить $\boldsymbol{n}^j$ в $\mathcal{P}$. Заменить $\boldsymbol{p}$ на $\boldsymbol{n}^j$ и вернуться к шагу 6.1.

Шаг 6.5. Если $\mathcal{N} = \varnothing$, то перейти к шагу 6.3, в противном случае перейти к шагу 6.6.

Шаг 6.6. Поиск пути заканчивается на брусе $\boldsymbol{p}$, который добавляется в множество концов путей $\mathcal{E}$. Перейти к шагу 7.

Шаг 7. Если в множестве $\mathcal{E}$ содержится $p_{attempts}$ элементов, перейти к шагу 8, в противном случае перейти к шагу 3.

Шаг 8. Определить вектор $m = \sum_{i=1}^{p_{attempts}} \mathrm{mid}(\boldsymbol{e}^i)$, где $\boldsymbol{e}^i \in \mathcal{E}$. Заменить брус $\boldsymbol{t}$ на брус $[m_1 - \tilde{w}; m_1 + \tilde{w}] \times \ldots \times [m_n - \tilde{w}; m_n + \tilde{w}] \cap \boldsymbol{t}, \tilde{w} = w \cdot \omega(\boldsymbol{t}) / 2$. Перейти к шагу 2.

Рекомендации по подбору параметров

Уменьшение параметра ε и увеличение параметров $w, p_{amount}, p_{attempts}, r_{\max}$ приводит к одновременному увеличению точности и времени работы алгоритма. Обратные действия приводят к общему ускорению, но за счет снижения точности работы алгоритма.

Следует обратить внимание, что на задачах большой размерности не рекомендуется задавать большие значения параметра p_{amount}, так как при описанной реализации работы алгоритма это приведет к переполнению памяти компьютера.

1.3.2. МЕТОД СТОХАСТИЧЕСКОЙ СЕТКИ

Стратегия поиска

Стратегия данного метода заключается в построении сетки на брусе поиска случайным образом. Она представляет собой разбиение бруса $\boldsymbol{s}$ на множество $\{\boldsymbol{n}^i\}_{i=1}^N$ непересекающихся брусов, таких что $\boldsymbol{s} = \bigcup_{i=1}^{N} \boldsymbol{n}^i$. На каждом элементе сетки ищется значение естественного интервального расширения минимизируемой функции (в этом заключается основное отличие от метода среднего пути, где значение естественного интервального расширения рассматривается не на всех элементах сетки). Такая процедура повторяется несколько раз. В итоге исходная область поиска разбивается на подобласти, каждая из которых имеет свой вес, подсчитанный на основе полученных значений естественного интервального расширения минимизируемой функции. Из подобластей выбирается самая перспективная (с большим весом), и алгоритм повторяется уже над ней.

Алгоритм

Шаг 1. Задать $\boldsymbol{s}$ – область поиска, ε – параметр точности (отвечает за размер выходного бруса), w – «скорость» уменьшения бруса, p_{amount} – количество точек для построения случайной сетки, $p_{attempts}$ – количество повторений. Положить целевой брус $\boldsymbol{t} = \boldsymbol{s}$.

Шаг 2. Если $\omega(\boldsymbol{t}) < \varepsilon$ – закончить работу алгоритма, вернув брус $\boldsymbol{t}$, в противном случае положить $a = 1$, множество брусов сетки $\mathcal{G} = \{\boldsymbol{g}^i\}_{i=1}^{l} = \{\boldsymbol{t}\}$, множество значений весов сетки $\mathcal{V} = \{v_i = 0\}_{i=1}^{l}$, перейти к шагу 3.

Шаг 3. Если $a \le p_{attempts}$, то перейти к шагу 4, в противном случае перейти к шагу 18.

Шаг 4. Положить величину $\Delta = \overline{\boldsymbol{f}(\boldsymbol{t})}$.

Шаг 5. Случайным образом, используя равномерное распределение, выбрать в целевом брусе p_{amount} точек.

Шаг 6. *Операция разбиения относительно выбранных случайных точек*. Разбить целевой брус относительно выбранных на шаге 5 точек на множество брусов $\mathcal{G} = \{\boldsymbol{g}^i\}_{i=1}^{\tilde{l}}$ (так как любой брус по сути является n-мерным параллелепипедом, то разбиение можно делать следующим образом: в каждой точке строятся n гиперплоскостей, каждая из которых

параллельна одной паре сторон данного параллелепипеда; совокупность этих гиперплоскостей и делит исходный брус), где $\tilde{l}$ – число полученных брусов.

Шаг 7. Создать множество $\mathcal{V} = \{\tilde{v}_i = \underline{\boldsymbol{f}(\boldsymbol{g}^i)}\}_{i=1}^{\tilde{l}}$.

Шаг 8. Создать множество $\mathcal{D} = \{d_i = 0\}_{i=1}^{\tilde{l}}$.

Шаг 8.1. Положить $j = 1$.

Шаг 8.2. Положить $k = 1$.

Шаг 8.3. Если $\tilde{v}_k - \tilde{v}_j > 0$, то $d_j = d_j + \tilde{v}_k - \tilde{v}_j$.

Шаг 8.4. Положить $k = k + 1$. Если $k = \tilde{l} + 1$, то $d_j = d_j + \Delta - \tilde{v}_{j^*}$, где $j^* = \operatorname{Arg}\max_j \tilde{v}_j$, и перейти к шагу 8.4. Если $k = \tilde{l} + 2$, то перейти к шагу 8.5. Если ни одно из предыдущих условий не выполнено, то перейти к шагу 8.3.

Шаг 8.5. Положить $j = j + 1$. Если $j = \tilde{l} + 1$, то перейти к шагу 9, в противном случае перейти к шагу 8.2.

Шаг 9. Создать множество брусов $\mathcal{N} = \varnothing$ и множество значений $\mathcal{NV} = \varnothing$. Положить $id = 1$.

Шаг 10. Положить $j = 1$.

Шаг 11. Положить $k = 1$.

Шаг 12. Добавить в множество $\mathcal{N}$ брус $\boldsymbol{g}^j \cap \boldsymbol{g}^k$, в множество $\mathcal{NV}$ значение $nv_{id} = v_j + d_k / \sum_{k=1}^{\tilde{l}} d_k$, положить $id = id + 1$.

Шаг 13. Положить $k = k + 1$. Если $k = \tilde{l} + 1$, то перейти к шагу 14, в противном случае – к шагу 12.

Шаг 14. Положить $j = j + 1$. Если $j = \tilde{l} + 1$, то перейти к шагу 15, в противном случае - к шагу 12.

Шаг 15. Положить интервал $\boldsymbol{m} =]+\infty; +\infty[$.

Шаг 16. Для каждого из брусов $\boldsymbol{n}^j \in \mathcal{N}$ выполнить:

- если $\overline{\boldsymbol{f}(\boldsymbol{n}^i)} < \underline{\boldsymbol{m}}$, то положить множества $\mathcal{G} = \{\boldsymbol{n}^i\}, \mathcal{V} = \{nv_i\}$ и $\boldsymbol{m} = \boldsymbol{f}(\boldsymbol{n}^i)$;
- если $\underline{\boldsymbol{f}(\boldsymbol{n}^i)} < \overline{\boldsymbol{m}}$, то продолжить перебор;
- если ни одно из предыдущих условий не выполнено, то добавить $\boldsymbol{n}^i$ в $\mathcal{G}$ и nv_i в $\mathcal{V}$, положить $\boldsymbol{m} = \boldsymbol{m} \cup \boldsymbol{f}(\boldsymbol{n}^i)$.

Шаг 17. Положить $a = a + 1$. Перейти к шагу 3.

Шаг 18. Положить $\boldsymbol{t} = [m_1 - \tilde{w}; m_1 + \tilde{w}] \times \ldots \times [m_n - \tilde{w}; m_n + \tilde{w}] \cap \boldsymbol{t}, m = \mathrm{mid}(\boldsymbol{g}^{i^*})$, $\tilde{w} = v \cdot \omega(\boldsymbol{t}) / 2$, $i^* = \mathrm{Arg}\max_i \tilde{v}_i$, перейти к шагу 2.

Рекомендации по подбору параметров

Уменьшение параметра ε и увеличение параметров $w, p_{amount}, p_{attempts}$ приводит к одновременному увеличению точности и времени работы алгоритма. Обратные действия приводят к общему ускорению за счет снижения точности работы алгоритма.

Следует обратить внимание, что на задачах большой размерности не рекомендуется задавать большие значения параметра p_{amount}, т.к. при описанной реализации работы алгоритма это приведет к переполнению памяти.

1.3.3. МЕТОД ИНТЕРВАЛЬНОГО РАЗБРОСАННОГО ПОИСКА

Стратегия поиска

В основе метода интервального разбросанного поиска лежат пять операций:

- метод генерации разнообразных решений (diversification generation method) – генерирует множество разнообразных решений;
- метод улучшения решений (improvement method) – преобразование решения в несколько других решений;
- метод обновления множества элементарных исходов (reference set update method) – обновление множества элементарных исходов новыми сгенерированными решениями;
- метод генерации подмножеств (subset generation method) – генерация подмножеств из множества элементарных исходов;
- метод комбинации решений (solution combination method) – преобразование подмножества элементарных исходов в несколько комбинированных решений.

Метод работает следующим образом: генерируются начальные решения (брусы), из которых создается множество элементарных исходов. Далее начинается итеративная часть (образующая цикл): генерация подмножеств, комбинация решений, улучшение полученных решений, обновление множества элементарных исходов. Итерации заканчиваются, когда за цикл множество элементарных исходов не обновилось ни одним новым элементом.

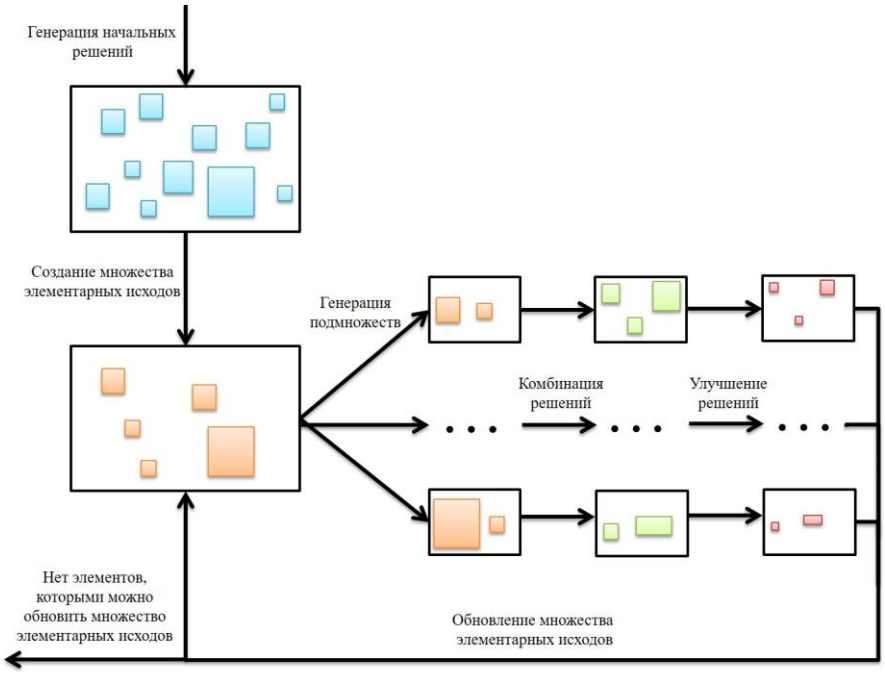


Рис. 1.4. Схема метода интервального разбросанного поиска

Алгоритм

Шаг 1. Задать брус поиска $\boldsymbol{s}$, параметр точности ε (отвечает за размер выходного бруса), размер множества начальных решений p_{size}, долю ширины w, размер множества элементарных исходов rs_{size}, доля лучших брусов b, размер подмножества sub_{size}, параметр способа разбиения $\alpha = 0$ или $\alpha = 1$.

Шаг 2. *Метод генерации разнообразных решений*.

Шаг 2.1. Наполнить множество $\mathcal{P}$ брусами $[m_1 - \tilde{w}; m_1 + \tilde{w}] \times \ldots \times [m_n - \tilde{w}; m_n + \tilde{w}] \cap \boldsymbol{s}, \tilde{w} = w \cdot \omega(\boldsymbol{s}) / 2$, где m_i – случайная точка из i-й компоненты бруса $\boldsymbol{s}$.

Шаг 2.2. Отсортировать $\mathcal{P}$ по возрастанию величины $\underline{\boldsymbol{f}(\boldsymbol{p}^i)}$, где $\boldsymbol{p}^i \in \mathcal{P}$.

Шаг 3. *Метод конструирования множества элементарных исходов*.

Шаг 3.1. Добавить в множество $\mathcal{RS}$ $\langle b \cdot p_{size} \rangle$ ($\langle \cdot \rangle$ – означает целую часть от числа) первых элементов множества $\mathcal{P}$, где p_{size} – размер множества $\mathcal{P}$. Добавленные элементы убираются из множества $\mathcal{P}$.

Шаг 3.2. Создать множество $\mathcal{D} = \{d_i = \sum_{j=1}^{\langle b \cdot p_{size} \rangle} h_\infty(\boldsymbol{p}^i, \boldsymbol{rs}^j)\}_{i=1}^{\tilde{p}_{size}}$, где $\tilde{p}_{size}$ – количество оставшихся в $\mathcal{P}$ элементов, $\boldsymbol{rs}^j$ – элементы множества $\mathcal{RS}$. Упорядочить элементы множества $\mathcal{D}$ по убыванию.

Шаг 3.3. Добавить в множество $\mathcal{RS}$ $(rs_{size} - \langle b \cdot p_{size} \rangle)$ брусов из $\mathcal{P}$, которым соответствуют наибольшие значения из $\mathcal{D}$.

Шаг 4. Отсортировать множество $\mathcal{RS}$ по возрастанию величины $\underline{\boldsymbol{f}(\boldsymbol{rs}^i)}$, где $\boldsymbol{rs}^i \in \mathcal{RS}$.

Шаг 5. Если $\omega(\boldsymbol{rs}^1) \geq \varepsilon$, то перейти к шагу 6. В противном случае закончить работу алгоритма, вернув брус $\boldsymbol{rs}^1$.

Шаг 6. Метод генерации подмножеств. Создать множество $\mathcal{SUB}$, состоящее из всех подмножеств размера sub_{size} множества $\mathcal{RS}$.

Шаг 7. Создать множество комбинированных решений $\mathcal{CS} = \varnothing$.

Шаг 8. Метод комбинации решений.

Шаг 8.1. Для каждого подмножества $sub^i = \{\boldsymbol{s}_j^i \in \mathcal{RS}\}_{j=1}^{sub_{size}} \in \mathcal{SUB}$ создать брус $\boldsymbol{u}^i = \bigcup_{j=1}^{sub_{size}} \boldsymbol{s}_j^i$, положить величину $mw_i = \min \omega(\boldsymbol{s}_j^i)$.

Шаг 8.2. Рекурсивно бисектировать (см. разд. 1.2) брусы $\boldsymbol{u}^i$ до тех пор, пока омега-характеристика полученных брусов не будет меньшей или равной величине mw_i. Полученные брусы добавить в множество $\mathcal{CS}$.

Шаг 9. Создать множество улучшенных решений $\mathcal{IS} = \varnothing$.

Шаг 10. Метод улучшения решений.

Шаг 10.1. Если $\alpha = 0$, то перейти к шагу 10.2, в противном случае – к шагу 10.3.

Шаг 10.2. Для каждого из бруса $\boldsymbol{cs}^i \in \mathcal{CS}$, выполнить операцию разбиения относительно центральной точки (см. разд. 1.3.1). Полученные брусы добавить в множество $\mathcal{IS}$.

Шаг 10.3. Для каждого из брусов $\boldsymbol{cs}^i \in \mathcal{CS}$, выполнить операцию разбиения относительно случайной точки (см. разд. 1.3.2 и 1.3.3). Полученные брусы добавить в множество $\mathcal{IS}$.

Шаг 11. Метод обновления множества элементарных исходов. Для каждого бруса $\boldsymbol{is}^i \in \mathcal{IS}$, если есть такое число j, для которого выполнено соотношение $\underline{\boldsymbol{f}(\boldsymbol{is}^i)} + \omega(\boldsymbol{f}(\boldsymbol{is}^i)) < \underline{\boldsymbol{f}(\boldsymbol{rs}^j)} + \omega(\boldsymbol{f}(\boldsymbol{rs}^j))$, то вставить перед j-м брусом в множестве $\mathcal{RS}$ брус $\boldsymbol{is}^i$, а последний элемент множества $\mathcal{RS}$ удалить.

Шаг 12. Если на шаге 11 в множестве $\mathcal{RS}$ произошли изменения, перейти к шагу 5, в противном случае – закончить работу алгоритма, вернув брус $\boldsymbol{rs}^1$.

Рекомендации по подбору параметров

Уменьшение параметра ε и увеличение параметров $p_{size}, rs_{size}, sub_{size}$ приводит к одновременному увеличению точности и времени работы алгоритма. Обратные действия приводят к общему ускорению расчетов, но за счет снижения точности работы алгоритма.

1.3.4. ИНТЕРВАЛЬНЫЙ ГЕНЕТИЧЕСКИЙ АЛГОРИТМ

Стратегия поиска

Генетические алгоритмы являются эвристическими алгоритмам поиска, которые основываются на свойствах процессов естественной эволюции [41, 43]. В основе этих алгоритмов лежат наследование, мутация, отбор и скрещивание. При описании используются определения, используемые в генетике: популяция – конечное множество особей (особи, которые входят в популяции, представляются хромосомами с закодированными в них множествами параметров), хромосомы – упорядоченные последовательности генов, ген – атомарный элемент генотипа, в частности, хромосомы, генотип – набор хромосом данной особи, фенотип – набор значений, соответствующих данному генотипу, т.е. декодированная структура или множество параметров.

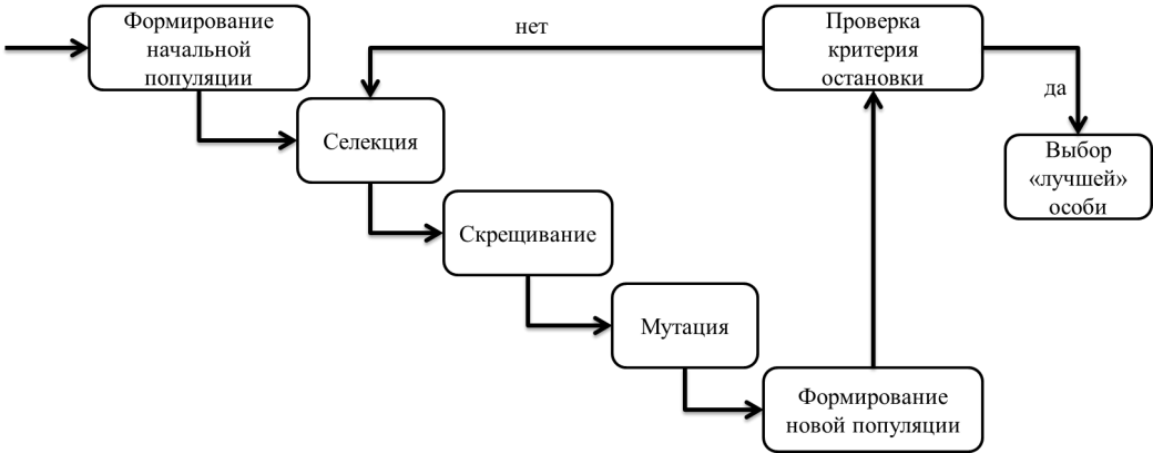


Рис. 1.5. Схема генетического алгоритма

В интервальном генетическом алгоритме с помощью вектора генов (генотипа) кодируется брус, который ассоциирован с отдельной особью. В разработанном алгоритме могут использоваться два вида кодирования: модифицированное бинарное, где каждая компонента бруса представляется в виде последовательностей нулей и единиц, и тернарное (LR), использующее систему из трех символов.

Бинарное кодирование. Если в классическом алгоритме порядковый номер исследуемой точки кодируется путем перевода номера в двоичную систему исчисления, то в разработанном алгоритме для кодирования бруса предлагается разбить каждый из интервалов, образующих область поиска на подынтервалы. Единица в гене указывает на то,

что в закодированном брусе данный подынтервал присутствует. Важным является тот факт, что последовательность подряд идущих единиц должна быть единственной.

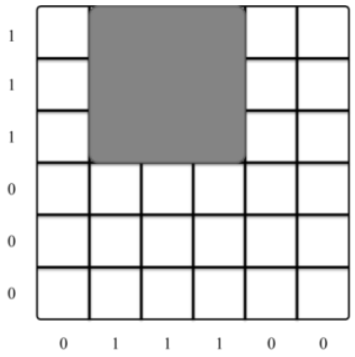


Рис. 1.6. Брус, соответствующий коду 011100×000111

Тернарное (LR) кодирование. Предлагается кодировать брус последовательностью символов «L», «R», которые будут указывать, какая часть бруса будет браться после применения операции бисекции, и «S», который будет указывать, когда прекращать бисекцию брус. Условимся, что $L(\boldsymbol{x}) = \boldsymbol{x}_1 \times \ldots \times [\underline{\boldsymbol{x}_i}; \mathrm{mid}(\boldsymbol{x}_i)] \times \ldots \times \boldsymbol{x}_n$, $R(\boldsymbol{x}) = \boldsymbol{x}_1 \times \ldots \times [\mathrm{mid}(\boldsymbol{x}_i); \overline{\boldsymbol{x}_i}] \times \ldots \times \boldsymbol{x}_n$, где i – номер первого интервала, для которого $\omega(\boldsymbol{x}) = \omega(\boldsymbol{x}_i)$. Таким образом, шаблон, которым представляется последовательность кодирующих символов, выглядит следующим образом: $A_1 A_2 \ldots A_m S_{m+1} S_{m+2} \ldots S_n$, где вместо символа «$A_i$», может быть «$L$» или «$R$». При декодировании происходит следующее: первым шагом декодированный брус $\boldsymbol{d}$ приравнивается брусу $\boldsymbol{s}$, представляющему область поиска. Далее просматриваются все символы в кодирующей строке, до первого появления «S». Если текущий символ «L», то $\boldsymbol{d}$ заменяется на $L(\boldsymbol{d})$, если «R» – на $R(\boldsymbol{d})$. Такой способ кодирования имеет ряд достоинств: уменьшается длина хромосомы и упрощается реализация генетических операторов.

Первым шагом алгоритма является формирование начальной популяции, в ходе которой случайным образом в соответствии с выбранным способом кодирования генерируются генотипы каждой из особей с учетом особенностей описанных ранее шаблонов.

Для выполнения селекции необходимо оценить каждую из особей. Для этого используется понятие «функции приспособленности». Для данного алгоритма она была составлена следующим образом:

$$F(\boldsymbol{x}) = \frac{\overline{\boldsymbol{f}(\boldsymbol{s})} - \underline{\boldsymbol{f}(\boldsymbol{x})}}{\omega(\boldsymbol{f}(\boldsymbol{s}))}. \qquad (1.4)$$

Выбор данной функции приспособленности обосновывается эвристическим правилом, что особь считается лучше (обладает большим значением функции приспособленности), если нижняя граница значения естественного интервального расширения минимизируемой функции меньше.

Селекция. Так как используемая функция приспособленности каждому брусу ставит в соответствие некоторое вещественное число, то в качестве операторов селекции можно использовать классические операторы панмиксии (популяция формируется случайным образом), рулетки и турнирной селекции [82].

Скрещивание. Для скрещивания было выбрано одноточечное скрещивание [82]. Кроме того, был создан оператор случайного сегмента (только для бинарного скрещивания): у хромосом родительских особей определяются минимальный и максимальный номера положений единиц, генерируются два случайных числа в промежутке между выбранными ранее двумя значениями. Все позиции слева от наименьшего числа и справа от наибольшего заполняются нулями, остальные – единицами.

Операция разделения. Так как при использовании этого типа скрещивания при бинарном кодировании может получиться хромосома, содержащая несколько последовательностей подряд идущих единиц, что противоречит описанному способу кодирования, то после скрещивания происходит разделение: например, из строки «110001» получатся две строки: «110000» и «000001».

Операция приведения. При использовании LR – кодирования также может возникнуть ситуация, когда получается хромосома, отличающаяся от шаблона. В этом случае необходимо выполнять операцию приведения. Для этого определяется минимальный номер j, такой что после него все символы в коде это «S». Далее все символы «S», которые находятся до j, заменяются случайным образом на «L» или «R».

Мутация. В качестве операторов мутации были выбраны: одноточечная мутация (в случайной хромосоме на случайной позиции происходит изменение значения), перемешивание (хромосомы выстраиваются в случайном порядке), инверсия (в одной хромосоме выбранный участок выстраивается в обратном порядке), транслокация (в границах одной хромосомы случайно выбранный кусочек перемещается на случайно выбранную позицию). Так как в процессе мутации может получиться хромосома, представляемая некорректной строкой, то на этом этапе так же необходимо проводить операцию разделения или приведения в зависимости от типа кодирования.

Формирование новой популяции может проводиться в соответствии с правилом простого усечения (старая популяция и мутированные потомки сливаются, сортируются по

убыванию значения функции приспособленности особей, полученное множество особей усекается до максимально возможного размера, который определяется при инициализации алгоритма) или случайным образом (из старой популяции и мутированных потомков случайно выбираются особи до тех пор, пока популяция не достигнет максимального размера).

Алгоритм

Шаг 1. Задать номер итерации алгоритма $t = 0$, максимальное количество итераций $t_{\max}$; число ε, отвечающее за длину строки кодирования (отвечает за размер выходного бруса); pop_{amount} – размер популяции, $pool_{amount}$ – размер родительского пула, r – вероятность мутации; $\mathbf{s}$ – область поиска.

Шаг 2. *Создание начальной популяции*.

Бинарное кодирование. Вначале необходимо определить длину кодирующей строки. Для этого находятся наименьшие из возможных числа n_i, такие что $\frac{\omega(\mathbf{s}_i)}{n_i} \le \varepsilon$. Далее генерируются pop_{amount} особей популяции. Каждая особь представляет собой вектор, каждая компонента которого состоит из последовательности нулей и единиц. Каждая такая последовательность генерируется следующим образом: выбираются два случайных целых числа от 1 до n_i. Все позиции, слева от наименьшего из них и справа от наибольшего заполняются нулями, остальные – единицами.

LR – кодирование. Вначале необходимо определить длину кодирующей строки. Для этого находятся наименьшие из возможных числа n_i, такие что $\omega(\mathbf{s}_i) / 2^{n_i} \le \varepsilon$. Тогда длина кодирующей строки будет $N = \sum_i n_i$. Далее генерируются pop_{amount} особей популяции. Каждая особь представляет собой последовательность символов «L», «R» и «S». Каждая такая последовательность генерируется следующим образом: с равной вероятностью текущую позицию может занять любой символ. Если помещаемый символ – «S», то все последующий символы справа так же будут «S».

Шаг 3. *Селекция*.

В процессе селекции осуществляется набор особей в родительский пул. В алгоритме могут использоваться следующие виды селекции:

- панмиксия (случайный равновероятный отбор) – у всех особей равные шансы попасть в пул;

- рулетка – нужное количество особей выбирается путем $pool_{amount}$ запусков рулетки, колесо которой содержит по одному сектору для каждого члена популяции (размер сектора особи пропорционален величине ее приспособленности);
- турнир – случайным образом выбираются два претендента, из которых в пул добавляется тот, чье значение функции приспособленности выше.

Шаг 4. *Скрещивание*.

В процессе скрещивания особей-родителей порождаются новые особи-потомки, которые потом будут участвовать в формировании новой популяции. В алгоритме могут использоваться следующие виды скрещивания:

- одноточечное – случайным образом определяется точка разрыва, относительно которой разрываются обе особи-родители, и соответствующие родительские сегменты «склеиваются»;
- случайное сегментирование - у хромосом родительских особей определяются минимальный и максимальный номера положений единиц, генерируются два случайных числа в промежутке между выбранными ранее двумя значениями (все позиции слева от наименьшего числа и справа от наибольшего заполняются нулями, остальные – единицами).

Выполнить операции разделения или приведения в зависимости от выбранного способа кодирования.

Шаг 5. *Мутация*.

В процессе мутации хромосомы особей могут подвергнуться изменениям. Это делается для того, чтобы особи были разнообразны. Для каждой особи генерируется случайное число $\xi \in [0;1]$. Если $\xi \le r$ – происходит мутация. В алгоритме могут использоваться следующие виды мутации:

- одноточечная;
- перемешивание;
- транслокация;
- инверсия.

Выполнить операции разделения или приведения в зависимости от выбранного способа кодирования.

Шаг 6. *Формирование новой популяции*.

Все потомки добавляются в исходную популяцию. Далее формирование может проводиться по следующим правилам:

- случайное – из популяции удаляются случайные особи, пока размер популяции не будет равен pop_{amount};
- простое усечение – особи сортируются по убыванию соответствующих значений функции приспособленности, и удаляется «хвост» популяции, чтобы осталось pop_{amount} особей.

Шаг 7. *Дробление* (только для LR - кодирования).

Для каждой особи выполнить следующее: если в хромосоме есть хотя бы один символ «S», то заменить первый из них на случайно выбранный символ «L» или «R».

Шаг 8. Увеличить t на единицу. Если $t = t_{\max}$, то вернуть брус, который закодирован в особи с наилучшим значением функции приспособленности. В противном случае – перейти к шагу 3.

Рекомендации по подбору параметров

Уменьшение параметра ε и увеличение параметров $t_{\max}, pop_{amount}, pool_{amount}, r$ приводит к одновременному увеличению точности и времени работы алгоритма. Обратные действия приводят к общему ускорению расчетов, но за счет снижения точности работы алгоритма.

1.3.5. ИНТЕРВАЛЬНЫЙ МЕТОД ВЗРЫВОВ

Стратегия поиска

На первом этапе метода случайным образом на области поиска устанавливаются бомбы, каждая из которых ассоциируется с некоторым брусом, описывающим ее местоположение. Кроме этого, бомбы сортируются по возрастанию нижней границы значения естественного интервального расширения минимизируемой функции. В алгоритме есть два итерационных этапа: глобального поиска и уточняющего, которые отличаются процедурой взрыва. В первом случае реализуется взрыв по всем координатам, а во втором – по одной из координат. На итерационных этапах алгоритма происходит расчет мощностей бомб; реализация взрыва, в ходе которого каждая из бомб образует осколки, которые разлетаются в разные стороны; обновление списка бомб. Алгоритм заканчивает работу, когда превышено максимальное количество итераций. Из списка бомб выбирается та, которой соответствует наименьшая нижняя граница значения естественного интервального расширения минимизируемой функции.

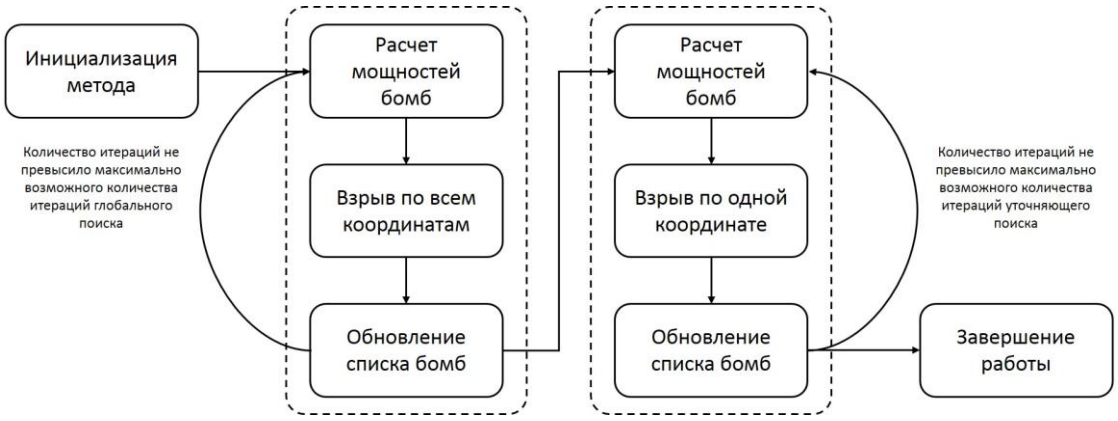


Рис. 1.7. Схема интервального метода взрывов

Алгоритм

Шаг 1. Задать номер итерации алгоритма $t^g = 0$, максимальное количество итераций глобального и уточняющего поисков $t^g_{\max}, t^c_{\max}$ (отвечают за размер выходного бруса), максимальное количество бомб $B_{\max}$, вектор максимальных мощностей $r_{\max} \in \mathbb{R}^n$ и область поиска $\boldsymbol{s}$.

Шаг 2. Случайным образом создать множество бомб-брусов: $\{\boldsymbol{b}^i = [\xi^i_1; \zeta^i_1] \times \ldots \times [\xi^i_n; \zeta^i_n]\}_{i=1}^{B_{\max}}$, где ξ^i_j, ζ^i_j - случайные величины, равномерно распределенные на интервале $\boldsymbol{s}_j$ (если $\xi^i_j > \zeta^i_j$, то соответствующий интервал заменяется на $[\zeta^i_j; \xi^i_j]$). Отсортировать бомбы по возрастанию нижней границы значения естественного интервального расширения минимизируемой функции.

Шаг 3. *Расчет мощностей бомб*. Для каждой бомбы $\boldsymbol{b}^i$ рассчитать соответствующую мощность: $p^i = \dfrac{i-1}{B_{\max}-1} \cdot r_{\max}, i = 1, \ldots, B_{\max}$.

Шаг 4. *Взрыв* (*глобальный поиск*). Во время этого этапа происходит образование двух осколков (операция бисекции бруса вдоль компоненты с номером k, обладающей наибольшей шириной). Таким образом, получаются две новых бомбы:

$$\boldsymbol{b}^i = \begin{pmatrix} \begin{pmatrix} \boldsymbol{b}^i_1 \\ \vdots \\ \boldsymbol{b}^i_k \\ \vdots \\ \boldsymbol{b}^i_n \end{pmatrix} + \begin{pmatrix} \xi(-p^i_1, p^i_1) \\ \vdots \\ \xi(-p^i_k, 0) \\ \vdots \\ \xi(-p^i_n, p^i_n) \end{pmatrix} \end{pmatrix} \cap \boldsymbol{s} \text{ и } \boldsymbol{b}^{B_{\max}+i} = \begin{pmatrix} \begin{pmatrix} \boldsymbol{b}^i_1 \\ \vdots \\ \boldsymbol{b}^i_k \\ \vdots \\ \boldsymbol{b}^i_n \end{pmatrix} + \begin{pmatrix} \xi(-p^i_1, p^i_1) \\ \vdots \\ \xi(0, p^i_k) \\ \vdots \\ \xi(-p^i_n, p^i_n) \end{pmatrix} \end{pmatrix} \cap \boldsymbol{s}, \quad (1.5)$$

где $\xi(a,b) = [\xi^b_a; \xi^b_a]$, ξ^b_a – случайная величина, равномерно распределенная на интервале $[a;b]$.

Шаг 5. *Обновление списка бомб*. Так как на предыдущем этапе количество бомб удвоилось, лишние необходимо удалить. Для этого производится сортировка бомб по возрастанию нижней границы значения естественного интервального расширения минимизируемой функции. Далее просто удаляются последние $B_{\max}$ бомб.

Шаг 6. Увеличить t^g на единицу. Если $t^g < t^g_{\max}$, то перейти к шагу 3, в противном случае – к шагу 7.

Шаг 7. Положить $t^c = 0$.

Шаг 8. *Расчет мощностей бомб*. Для каждой бомбы $\boldsymbol{b}^i$ рассчитать соответствующую мощность: $p^i = \dfrac{i-1}{B_{\max}-1} \cdot r_{\max}, i = 1,\ldots, B_{\max}$.

Шаг 9. *Взрыв* (*уточняющий поиск*). В ходе данного этапа получаются две новых бомбы:

$$\boldsymbol{b}^i = \left(\begin{pmatrix} \boldsymbol{b}^i_1 \\ \vdots \\ \boldsymbol{b}^i_k \\ \vdots \\ \boldsymbol{b}^i_n \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ \xi(-p^i_k, 0) \\ \vdots \\ 0 \end{pmatrix} \right) \cap \boldsymbol{s} \quad \text{и} \quad \boldsymbol{b}^{B_{\max}+i} = \left(\begin{pmatrix} \boldsymbol{b}^i_1 \\ \vdots \\ \boldsymbol{b}^i_k \\ \vdots \\ \boldsymbol{b}^i_n \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ \xi(0, p^i_k) \\ \vdots \\ 0 \end{pmatrix} \right) \cap \boldsymbol{s}, \tag{1.6}$$

где $\xi(a,b) = [\xi^b_a; \xi^b_a]$, ξ^b_a – случайная величина, равномерно распределенная на интервале $[a;b]$.

Шаг 10. *Обновление списка бомб*. Так как на предыдущем этапе количество бомб удвоилось, лишние необходимо удалить. Для этого производится сортировка бомб по возрастанию нижней границы значения естественного интервального расширения минимизируемой функции. Далее просто удаляются последние $B_{\max}$ бомб.

Шаг 11. Увеличить t^c на единицу. Если $t^c < t^c_{\max}$, то перейти к шагу 8, в противном случае – к шагу 12.

Шаг 12. Выбрать из всех бомб ту, которой соответствует наименьшее значение нижней границы оценки значения естественного интервального расширения минимизируемой функции.

Рекомендации по подбору параметров

Увеличение параметров $t^g_{\max}, t^c_{\max}, B_{\max}$ приводит к одновременному увеличению точности и времени работы алгоритма. Обратные действия приводят к общему ускорению расчетов, но за счет снижения точности работы алгоритма.

1.3.6. АДАПТИВНЫЙ ИНТЕРВАЛЬНЫЙ АЛГОРИТМ

Стратегия поиска

Адаптивный интервальный алгоритм в ходе своей работы оперирует тремя группами брусов, каждая из которых имеет свое коллективное знание и способ его обработки и использования. Под коллективным знанием подразумевается суммарная информация о целевой функции, накопленная в ходе работы алгоритма и представленная особым образом [53]. В алгоритме используются следующие группы брусов:

- *Best* – группа наиболее перспективных брусов (им соответствует наименьшее значение нижней границы естественного интервального расширения минимизируемой функции); коллективное знание данной группы представляется множеством брусов $Z_B = \{\boldsymbol{z}_B^i\}_{i=1}^{C_B^+ + C_B^-}$, в котором хранятся C_B^+ хороших и C_B^- плохих брусов, которые соответствуют наименьшим и наибольшим значениям нижней границы естественного интервального расширения минимизируемой функции соответственно; новый брус генерируется таким образом, что он «притягивается» к хорошим брусам и «отталкивается» от плохих;
- *Worst* – группа наименее перспективных брусов (им соответствует наибольшее значение нижней границы естественного интервального расширения минимизируемой функции); коллективное знание данной группы представляется в виде набора плотностей вероятности $p_i(x_i)$, $i = \overline{1,n}$, с помощью которых генерируются новые брусы: перспективные области (с меньшим значением нижней границы естественного интервального расширения минимизируемой функции) имеют большую вероятность быть выбранными;
- *Average* – группа брусов, не вошедших в предыдущие группы; коллективное знание данной группы представляется множеством брусов $Z_A = \{\boldsymbol{z}_A^i\}_{i=1}^{C_A}$; новый брус генерируется таким образом, чтобы его суммарное расстояние от брусов множества Z_A было наибольшим; таким образом стимулируется исследование новых областей.

Свойство адаптивности данного алгоритма реализуется за счет постоянного анализа коллективного знания каждой группы, т.е. использования особенностей целевой функции и применения различных механизмов генерации новых брусов.

Алгоритм

Шаг 1. Задать параметры метода:

- общие параметры алгоритма: $N_{\max} \in \mathbb{N}$ – максимальное количество итераций (параметр отвечает за размер выходного бруса), $\gamma \in (0;1)$ – коэффициент сжатия, $w_{init} \in [0;1]$ – коэффициент генерации начальной ширины, $\boldsymbol{s}$ – область поиска,
- параметры группы *Best*: $N_B \in \mathbb{N}$ – максимальный размер группы *Best*, $\gamma_B \in (0;1)$ – коэффициент сжатия, $C_B^+, C_B^- \in \mathbb{N}$ – количество хороших и плохих брусов для корректировки соответственно, $\alpha_B, \beta_B \in (0;1)$ – коэффициенты влияния лучших и худших брусов из коллективного знания соответственно (при этом $\alpha_B + \beta_B \in (0;1)$), $r^B = (r_1^B, r_2^B, ..., r_n^B)^s \in (\mathbb{R}^+)^n$ – вектор возможных значений шага, $q_i^+ \in (0;1)$, $i = 1,..., C_B^+$, $q_j^- \in (0;1)$, $j = 1,..., C_B^-$ – доли влияния хороших и плохих брусов соответственно ($\sum_{i=1}^{C_B^+} q_i^+ = 1$ и $\sum_{i=1}^{C_B^-} q_i^- = 1$),
- параметры группы *Worst*: $N_W \in \mathbb{N}$ – максимальный размер группы *Worst*, $Sp_i \in \mathbb{N}$, $i = 1,..., n$ – количество интервалов разбиения i-й координаты бруса $\boldsymbol{s}$, $\rho_W \in (0;1)$ – коэффициент изменения шанса, $\gamma_W \in (0;1)$ – коэффициент сжатия, $A_{good} \in \mathbb{N}$, $A_{bad} \in \mathbb{N}$ – количества учитываемых хороших и плохих брусов соответственно,
- параметры группы *Average*: $N_A \in \mathbb{N}$ – максимальный размер группы *Average*, $\gamma_A \in (0;1)$ – коэффициент сжатия, $A_{\max} \in \mathbb{N}$ – число генерируемых брусов, $r^A = (r_1^A, r_2^A, ..., r_n^A)^{\mathrm{T}} \in (\mathbb{R}^+)^n$ – вектор возможных значений шага.

Шаг 2. Инициализация алгоритма.

Шаг 2.1. Инициализация групп.

Шаг 2.1.1. Положить $P = \{\boldsymbol{p}^i\} = \varnothing$ и $i = 1$.

Шаг 2.1.2. Сгенерировать случайную точку $m = (m_1, m_2, ..., m_n)^{\mathrm{T}}$, используя равномерный закон распределения на компонентах бруса $\boldsymbol{s}$.

Шаг 2.1.3. Используя сгенерированную точку, создать брус $\boldsymbol{p}^i = [m_1 - \tilde{w}; m_1 + \tilde{w}] \times ... \times [m_n - \tilde{w}; m_n + \tilde{w}], \tilde{w} = w_{init} \cdot \omega(\boldsymbol{s}) / 2$.

Шаг 2.1.4. Выполнить процедуру восстановления бруса $\boldsymbol{p}^i$ относительно бруса $\boldsymbol{s}$ (см. далее).

Шаг 2.1.5. Положить $i = i+1$. Если $i > N_B + N_A + N_W$, то перейти к шагу 2.1.2, в противном случае – к шагу 2.1.6.

Шаг 2.1.6. Отсортировать множество $P = \{\boldsymbol{p}^i\}_{i=1}^{N_B+N_A+N_W}$ по возрастанию в соответствии с величиной $\underline{\boldsymbol{f}(\boldsymbol{p}^i)}$.

Шаг 2.1.7. Положить $G_{best} = \{\boldsymbol{g}^i_{best}\}_{i=1}^{N_B} = \{\boldsymbol{p}^i\}_{i=1}^{N_B}$, $G_{average} = \{\boldsymbol{g}^i_{average}\}_{i=1}^{N_A} = \{\boldsymbol{p}^i\}_{i=N_B+1}^{N_B+N_A}$, $G_{worst} = \{\boldsymbol{g}^i_{worst}\}_{i=1}^{N_W} = \{\boldsymbol{p}^i\}_{i=N_B+N_A+1}^{N_B+N_A+N_W}$.

Шаг 2.2. Инициализация коллективного знания.

Шаг 2.2.1. Инициализация коллективного знания группы *Best*. Положить $Z_B = \{\boldsymbol{z}^i_B\} = \varnothing$.

Шаг 2.2.2. Инициализация коллективного знания группы *Average*. Положить $Z_A = \{\boldsymbol{z}^i_A\} = \varnothing$.

Шаг 2.2.3. Инициализация коллективного знания группы *Worst*. Положить величину $pr_i^j = 1$, где pr_i^j – шанс выбора в i-й компоненте бруса $\boldsymbol{s}$ j-го подынтервала, т.е. интервала $\boldsymbol{z}_i^j = [\underline{\boldsymbol{s}_i} + \frac{j-1}{Sp_i}\cdot\omega(\boldsymbol{s}_i); \underline{\boldsymbol{s}_i} + \frac{j}{Sp_i}\cdot\omega(\boldsymbol{s}_i)]$, $i = 1,...,n$, $j = 1,...,Sp_i$.

Шаг 2.3. Обновление коллективного знания.

Шаг 2.3.1. Обновить коллективное знание группы *Best* с помощью множества P.

Шаг 2.3.2. Обновить коллективное знание группы *Average* с помощью множества P.

Шаг 2.3.3. Обновить коллективное знание группы *Worst* с помощью множества P.

Шаг 3. Положить $iter = 1$.

Шаг 4. Обновление группы *Best*.

Шаг 4.1. Положить $i = 1$,

Шаг 4.2. Для бруса $\boldsymbol{g}^i_{best}$ с помощью коллективного знания сгенерировать брус $\tilde{\boldsymbol{g}}^i_{best}$.

Шаг 4.3. Сжать брус $\boldsymbol{g}^i_{best}$ относительно ширины $w_{best} = \frac{1}{n}\cdot\sum_{j=1}^{n}\omega(\boldsymbol{g}^i_{j,best})$.

Шаг 4.4. Если $\underline{\boldsymbol{f}(\tilde{\boldsymbol{g}}^i_{best})} < \underline{\boldsymbol{f}(\boldsymbol{g}^i_{best})}$, то заменить брус $\boldsymbol{g}^i_{best}$ на $\tilde{\boldsymbol{g}}^i_{best}$.

Шаг 4.5. Положить $i = i + 1$. Если $i > N_B$, то перейти к шагу 5, в противном случае – к шагу 4.2.

Шаг 5. Обновление группы *Average*.

Шаг 5.1. Положить $i = 1$,

Шаг 5.2. Для бруса $\boldsymbol{g}^i_{average}$ с помощью коллективного знания сгенерировать брус $\tilde{\boldsymbol{g}}^i_{average}$.

Шаг 5.3. Сжать брус $\boldsymbol{g}^i_{average}$ относительно ширины $w_{average} = \frac{1}{n} \cdot \sum_{j=1}^{n} \omega(\boldsymbol{g}^i_{j,average})$.

Шаг 5.4. Если $\underline{\boldsymbol{f}(\tilde{\boldsymbol{g}}^i_{average})} < \underline{\boldsymbol{f}(\boldsymbol{g}^i_{average})}$, то заменить брус $\boldsymbol{g}^i_{average}$ на $\tilde{\boldsymbol{g}}^i_{average}$.

Шаг 5.5. Положить $i = i + 1$. Если $i > N_A$, то перейти к шагу 6, в противном случае – к шагу 5.2.

Шаг 6. Обновление группы *Worst*.

Шаг 6.1. Положить $i = 1$,

Шаг 6.2. Для бруса $\boldsymbol{g}^i_{worst}$ с помощью коллективного знания сгенерировать брус $\tilde{\boldsymbol{g}}^i_{worst}$.

Шаг 6.3. Сжать брус $\boldsymbol{g}^i_{worst}$ относительно ширины $w_{worst} = \frac{1}{n} \cdot \sum_{j=1}^{n} \omega(\boldsymbol{g}^i_{j,worst})$.

Шаг 6.4. Если $\underline{\boldsymbol{f}(\tilde{\boldsymbol{g}}^i_{worst})} < \underline{\boldsymbol{f}(\boldsymbol{g}^i_{worst})}$, то заменить брус $\boldsymbol{g}^i_{worst}$ на $\tilde{\boldsymbol{g}}^i_{worst}$.

Шаг 6.5. Положить $i = i + 1$. Если $i > N_W$, то перейти к шагу 7, в противном случае – к шагу 6.2.

Шаг 7. Перегруппировка элементов групп *Best*, *Average*, *Worst*.

Шаг 7.1. Положить $P = \{\boldsymbol{p}^i\} = G_{best} \cup G_{average} \cup G_{worst}$.

Шаг 7.2. Отсортировать множество $P = \{\boldsymbol{p}^i\}_{i=1}^{N_B+N_A+N_W}$ по возрастанию в соответствии с величиной $\underline{\boldsymbol{f}(\boldsymbol{p}^i)}$.

Шаг 7.3. Положить $G_{best} = \{\boldsymbol{g}^i_{best}\}_{i=1}^{N_B} = \{\boldsymbol{p}^i\}_{i=1}^{N_B}$, $G_{average} = \{\boldsymbol{g}^i_{average}\}_{i=1}^{N_A} = \{\boldsymbol{p}^i\}_{i=N_B+1}^{N_B+N_A}$, $G_{worst} = \{\boldsymbol{g}^i_{worst}\}_{i=1}^{N_W} = \{\boldsymbol{p}^i\}_{i=N_B+N_A+1}^{N_B+N_A+N_W}$.

Шаг 8. Положить $P = \{\boldsymbol{p}^i\}_{i=1}^{N_B+N_A+N_W} = G_{best} \cup G_{average} \cup G_{worst}$.

Шаг 9. Обновление коллективного знания.

Шаг 9.1. Обновить коллективное знание группы *Best* с помощью множества P.

Шаг 9.2. Обновить коллективное знание группы *Average* с помощью множества P.

Шаг 9.3. Обновить коллективное знание группы *Worst* с помощью множества P.

Шаг 10. Положить $iter = iter + 1$. Если $iter > N_{\max}$, то перейти к шагу 11, в противном случае – к шагу 4.

Шаг 11. В качестве ответа вернуть брус $\boldsymbol{p}^* = \boldsymbol{g}^1_{best}$.

Замечания. В приведенном пошаговом алгоритме используются некоторые операции, подробно описанные ниже.

Алгоритм восстановления бруса $\boldsymbol{a}$ относительно бруса $\boldsymbol{b}$ (шаг 2.1.4).

Шаг 1. Положить $i = 1$.

Шаг 2. Если $\underline{\boldsymbol{a}_i} < \underline{\boldsymbol{b}_i}$, то положить $\underline{\boldsymbol{a}_i} = \underline{\boldsymbol{b}_i}$, перейти к шагу 3.

Шаг 3. Если $\overline{\boldsymbol{a}_i} < \underline{\boldsymbol{b}_i}$, то положить $\overline{\boldsymbol{a}_i} = \underline{\boldsymbol{b}_i}$, перейти к шагу 4.

Шаг 4. Если $\underline{\boldsymbol{a}_i} > \overline{\boldsymbol{b}_i}$, то положить $\underline{\boldsymbol{a}_i} = \overline{\boldsymbol{b}_i}$, перейти к шагу 5.

Шаг 5. Если $\overline{\boldsymbol{a}_i} > \overline{\boldsymbol{b}_i}$, то положить $\overline{\boldsymbol{a}_i} = \overline{\boldsymbol{b}_i}$, перейти к шагу 6.

Шаг 6. Положить $i = i + 1$. Если $i > n$, то закончить алгоритм, в противном случае перейти к шагу 2.

Алгоритм процедуры сжатия бруса $\boldsymbol{a}$ относительно ширины w (шаги 4.3, 5.3, 6.3).

Шаг 1. Найти среднюю точку бруса $\boldsymbol{a}$: $m = (m_1, ..., m_n)^{\mathrm{T}} = \mathrm{mid}(\boldsymbol{a})$.

Шаг 2. Используя найденную среднюю точку, создать брус $\boldsymbol{a} = [m_1 - \tilde{w}; m_1 + \tilde{w}] \times ... \times [m_n - \tilde{w}; m_n + \tilde{w}], \tilde{w} = \gamma \cdot w / 2$.

Шаг 3. Восстановить полученный брус $\tilde{\boldsymbol{a}}$ относительно бруса $\boldsymbol{a}$.

Шаг 4. Заменить брус $\boldsymbol{a}$ на $\tilde{\boldsymbol{a}}$.

Алгоритм обновления коллективного знания группы *Best* с помощью множества $P = \{\boldsymbol{p}^i\}_{i=1}^{N_B + N_A + N_W}$ (шаг 3.3.1).

Шаг 1. Положить $Z_B = \{\boldsymbol{z}_B^i\}_{i=1}^{C_B^+ + C_B^-} \cup P$.

Шаг 2. Отсортировать множество Z_B по возрастанию в соответствии с величиной $\underline{\boldsymbol{f}(\boldsymbol{z}_B^i)}$.

Шаг 3. Удалять из множества Z_B случайно выбранные брусы, пока выполнено условие $|Z_B| > C_B^+ + C_B^-$, где $|\cdot|$ – количество элементов множества.

Алгоритм обновления коллективного знания группы *Average* с помощью множества $P = \{\boldsymbol{p}^i\}_{i=1}^{N_B + N_A + N_W}$ (шаг 3.3.2).

Шаг 1. Положить $Z_A = \{\boldsymbol{z}_A^i\}_{i=1}^{C_A} \cup P$.

Шаг 2. Отсортировать множество Z_A по возрастанию в соответствии с величиной $\underline{\boldsymbol{f}(\boldsymbol{z}_A^i)}$.

Шаг 3. Удалять из множества Z_A случайно выбранные брусы, пока выполнено условие $|Z_A| > C_A$, где $|\cdot|$ – количество элементов множества.

Алгоритм обновления коллективного знания группы *Worst* с помощью множества $P = \{\boldsymbol{p}^i\}_{i=1}^{N_B+N_A+N_W}$ (шаг 3.3.3).

Шаг 1. Учет хороших брусов.

Шаг 1.1. Положить $id = 1$.

Шаг 1.2. Положить $i = 1$.

Шаг 1.3. Положить $j = 1$.

Шаг 1.4. Если $\boldsymbol{p}^i \cap \boldsymbol{z}_i^j \neq \varnothing$, то положить $pr_i^j = pr_i^j \cdot (1 + \rho_W^{id})$.

Шаг 1.5. Положить $j = j + 1$. Если $j > Sp_i$, то перейти к шагу 1.6, в противном случае – к шагу 1.4.

Шаг 1.6. Положить $i = i + 1$. Если $i > n$, то перейти к шагу 1.7, в противном случае – к шагу 1.3.

Шаг 1.7. Положить $id = id + 1$. Если $id > A_{good}$, то перейти к шагу 2, в противном случае – к шагу 1.2.

Шаг 2. Учет плохих брусов.

Шаг 2.1. Положить $id = 1$.

Шаг 2.2. Положить $i = 1$.

Шаг 2.3. Положить $j = 1$.

Шаг 2.4. Если $\boldsymbol{p}^{N_B+N_A+N_W-id+1} \cap \boldsymbol{z}_i^j \neq \varnothing$, то положить $pr_i^j = pr_i^j \cdot (1 - \rho_W^{id})$.

Шаг 2.5. Положить $j = j + 1$. Если $j > Sp_i$, то перейти к шагу 2.6, в противном случае – к шагу 2.4.

Шаг 2.6. Положить $i = i + 1$. Если $i > n$, то перейти к шагу 2.7, в противном случае – к шагу 2.3.

Шаг 2.7. Положить $id = id + 1$. Если $id > A_{good}$, то завершить работу, в противном случае перейти к шагу 2.2.

Алгоритм генерации бруса $\tilde{\boldsymbol{g}}_{best}^i$ для бруса $\boldsymbol{g}_{best}^i$ с помощью коллективного знания группы *Best* (шаг 4.2).

Шаг 1. Найти $w = \frac{1}{n} \cdot \sum_{j=1}^{n} \omega(\boldsymbol{g}_{j,best}^i)$ и среднюю точку бруса $\boldsymbol{g}_{best}^i$: $m = (m_1, ..., m_n)^{\mathrm{T}} = \mathrm{mid}(\boldsymbol{g}_{best}^i)$.

Шаг 2. Найти вектор случайного перемещения $\Delta^R = (\Delta_1^R, ..., \Delta_n^R)^T = (\xi_1 \cdot r_1^B, ..., \xi_n \cdot r_n^B)^{\mathrm{T}}$, где ξ_j, $j = 1, ..., n$ – случайная величина, равномерно распределенная на интервале $[-1;1]$.

Шаг 3. Найти вектор смещения в сторону хороших брусов $\Delta^G = (\Delta_1^G, ..., \Delta_n^G)^T = \sum_{j=1}^{C_B^+} q_j^+ \cdot (\mathrm{mid}(\boldsymbol{z}_B^j) - \mathrm{mid}(\boldsymbol{g}_{best}^i))$.

Шаг 4. Выполнить нормировку вектора сдвига: $\tilde{\Delta}^G = (\max\{1, \max_{j=1,...,n} |\Delta_j^G| / r_j^B\})^{-1} \cdot \Delta^G$.

Шаг 5. Найти вектор смещения в сторону плохих брусов $\Delta^B = (\Delta_1^B, ..., \Delta_n^B)^{\mathrm{T}} = \sum_{j=1}^{C_B^-} q_j^- \cdot (\mathrm{mid}(\boldsymbol{z}_B^{C_B^+ + C_B^- + 1 - j}) - \mathrm{mid}(\boldsymbol{g}_{best}^i))$.

Шаг 6. Выполнить нормировку вектора сдвига: $\tilde{\Delta}^B = (\max\{1, \max_{j=1,...,n} |\Delta_j^B| / r_j^B\})^{-1} \cdot \Delta^B$.

Шаг 7. Создать брус $\tilde{\boldsymbol{g}}_{best}^i$, j-я компонента которого рассчитывается по формуле: $[m_j + s - \gamma_B \cdot w/2; m_j + s + \gamma_B \cdot w/2], s = (1 - \alpha_B - \beta_B) \cdot \Delta_j^R + \alpha_B \cdot \tilde{\Delta}_j^G - \beta_b \cdot \tilde{\Delta}_j^B$.

Шаг 8. Выполнить процедуру восстановления бруса $\tilde{\boldsymbol{g}}_{best}^i$ относительно бруса $\boldsymbol{g}_{best}^i$.

Алгоритм генерации бруса $\tilde{\boldsymbol{g}}_{average}^i$ для бруса $\boldsymbol{g}_{average}^i$ с помощью коллективного знания группы *Average* (шаг 5.2).

Шаг 1. Найти $w = \frac{1}{n} \cdot \sum_{j=1}^{n} \omega(\boldsymbol{g}_{j,average}^i)$ и положить $T = \{\boldsymbol{t}^i\} = \varnothing$ и $id = 1$.

Шаг 2. Найти среднюю точку бруса $\boldsymbol{g}_{average}^i$: $m = (m_1, ..., m_n)^{\mathrm{T}} = \mathrm{mid}(\boldsymbol{g}_{average}^i)$

Шаг 3. Используя найденную ранее среднюю точку, создать брус $\boldsymbol{t}^{id} = [m_1 + \xi_1 \cdot r_1^A - \tilde{w}; m_1 + \xi_1 \cdot r_1^A + \tilde{w}] \times ... \times [m_n + \xi_n \cdot r_n^A - \tilde{w}; m_n + \xi_n \cdot r_n^A + \tilde{w}], \tilde{w} = \gamma_A \cdot w/2$, где ξ_j, $j = 1, ..., n$ – случайная величина, равномерно распределенная на интервале $[-1;1]$.

Шаг 4. Выполнить процедуру восстановления бруса $\boldsymbol{t}^{id}$ относительно бруса $\boldsymbol{g}_{average}^i$.

Шаг 5. Добавить брус $\boldsymbol{t}^{id}$ в T и положить $id = id + 1$. Если $id > A_{\max}$, то перейти к шагу 3, в противном случае – к шагу 6.

Шаг 6. Создать брус $\tilde{\boldsymbol{g}}_{average}^i = \mathrm{Arg} \max_{id=1,...,A_{max}} d_e(\mathrm{mid}(\boldsymbol{t}^i), \mathrm{mid}(\boldsymbol{z}_A^i))$, где $d_e(\cdot,\cdot)$ – расстояние между точками, $|\cdot|$ – количество элементов множества.

Алгоритм генерации бруса $\tilde{\boldsymbol{g}}_{worst}^i$ для бруса $\boldsymbol{g}_{worst}^i$ с помощью коллективного знания группы *Worst* (шаг 6.2):

Шаг 1. Положить $j = 1$.

Шаг 2. Случайным образом выбрать один из интервалов $\boldsymbol{z}_j^k$, $k = 1,...,Sp_j$, вероятность выбора которого вычисляется как $pr_j^k / \sum_{k=1}^{Sp_j} pr_j^k$ и сгенерировать случайную величину m_i, равномерно распределенную на нем.

Шаг 3. Положить $j = j + 1$. Если $j > n$, то перейти к шагу 4, в противном случае – к шагу 2.

Шаг 4. Найти $w = \frac{1}{n} \cdot \sum_{j=1}^{n} \omega(\boldsymbol{g}_{j,worst}^i)$.

Шаг 5. Создать брус $\tilde{\boldsymbol{g}}_{worst}^i = [m_1 - \tilde{w}; m_1 + \tilde{w}] \times ... \times [m_n - \tilde{w}; m_n + \tilde{w}], \tilde{w} = \gamma_W \cdot w / 2$.

Шаг 6. Выполнить процедуру восстановления бруса $\tilde{\boldsymbol{g}}_{worst}^i$ относительно бруса $\boldsymbol{g}_{worst}^i$.

Рекомендации по подбору параметров

Увеличение численных параметров алгоритма, отвечающих за размеры групп, приводит к повышению точности работы за счет снижения скорости работы. Влияние остальных параметров сложно формализовать, так как их действие зависит от особенностей оптимизируемой функции.

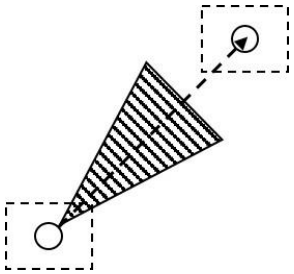
1.3.7. САМООРГАНИЗУЮЩИЙСЯ ИНТЕРВАЛЬНЫЙ АЛГОРИТМ ИМИТАЦИИ ЭВОЛЮЦИИ КОЛОНИИ БАКТЕРИЙ

Стратегия поиска

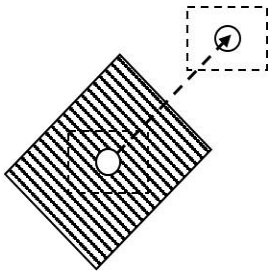
Предлагаемый алгоритм является самоорганизующимся интервальным алгоритмом глобальной условной оптимизации. В ходе своей работы он имитирует процесс эволюции колонии бактерий (каждая бактерия представляется брусом, описывающим область ее обитания), цель которого заключается в обеспечении всем особям колонии наилучшего места обитания (чем место лучше, тем меньше нижняя граница естественного интервального расширения минимизируемой функции).

Каждая бактерия отличается следующими параметрами:

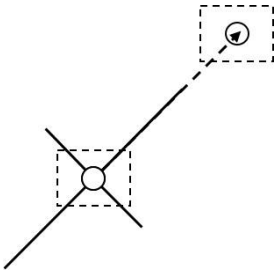
- способом передвижения; реализованы следующий типы передвижений:
  - «пирамидальный»



- «зональный»



- «звезда»



- стратегией передвижения (**жадная** стратегия, когда бактерия перемещается в первую область, которая лучше; **умная**, когда бактерия выбирает наилучшую из возможных областей; **цепная** стратегия, когда возможно несколько перемещений),
- параметрами, описывающими возможную область перемещения,
- текущим направлением движения,
- областью (местом) обитания, описываемой брусом.

Каждая итерация работы алгоритма представляет собой реализацию одного эволюционного цикла, в процессе которого бактерии колонии находят наилучшее место обитания и, заканчивая свой жизненный цикл, дают потомство (следующее поколение колонии).

В ходе процесса построения новой колонии используются следующий правила:

- 30% новой популяции составляют абсолютно случайные, заново сгенерированные бактерии,
- 30% новой популяции составляют бактерии, расположенные в областях обитания, найденных прошлым поколением,
- 30% новой популяции составляют бактерии, расположенные в той же начальной точке, что и прошлое поколение, но направление поиска противоположное,
- 10% новой популяции составляют бактерии, расположенные в наилучшем месте, найденном за все время работы алгоритма.

Для оценки места обитания применяется естественной интервальное расширение минимизируемой функции. Чем меньше значение нижней границы, тем лучше данная область.

Реализация данных правил позволяет поддерживать необходимый уровень «случайности» и наследования полезной информации, найденной предыдущим поколением.

Алгоритм

Шаг 1. Задать $N_{\max} \in \mathbb{N}$ – максимальное количество итераций, точность w_ε (отвечает за размер выходного бруса) и область поиска $\boldsymbol{s}$.

Шаг 2. Инициализация алгоритма.

Шаг 2.1. Положить $A_{\max} = 10 \cdot n$ – размер популяции и $G_{\max} = 10 \cdot n$ – максимальное значение параметра движения бактерий.

Шаг 2.2. Положить величину шага $P_{move} = (p^1_{move}, ..., p^n_{move})^T$, где $p^i_{move} = \omega(\boldsymbol{s}_i) / n$.

Шаг 2.3. Положить текущее лучшее значение $bv = +\infty$ и текущий лучший брус $\boldsymbol{bp} = \varnothing$.

Шаг 2.4. Генерация начальной популяции.

Шаг 2.4.1. Положить $id = 1$.

Шаг 2.4.2. Сгенерировать вектор, описывающий местоположение, $m^{id} = (m^{id}_1, ..., m^{id}_n)^T$, где $m_i \sim U(\underline{\boldsymbol{s}_i}, \overline{\boldsymbol{s}_i})$, $i = 1, ..., n$, $U(\cdot, \cdot)$ – равномерное распределение.

Шаг 2.4.3. Сгенерировать вектор, описывающий ширину, $w^{id} = (w^{id}_1, ..., w^{id}_n)^{\mathrm{T}} = (|v_1| \cdot \omega(\boldsymbol{s}_1), ..., |v_n| \cdot \omega(\boldsymbol{s}_n))^{\mathrm{T}}$, где $v_i \sim N(0, 0.25^2)$, $i = 1, ..., n$, $N(\cdot, \cdot)$ – нормальный закон распределения.

Шаг 2.4.4. Используя сгенерированную точку, создать брус $\boldsymbol{b}^{id}_l = \left\langle [m^{id}_1 - w^{id}_1 / 2; m^{id}_1 + w^{id}_1 / 2] \times ... \times [m^{id}_n - w^{id}_n / 2; m^{id}_n + w^{id}_n / 2] \right\rangle$, где $\langle \cdot \rangle$ – процедура восстановления (см. разд. 1.3.7).

Шаг 2.4.5. Сгенерировать вектор, описывающий направление, $d^{id} = (d^{id}_1, ..., d^{id}_n)^{\mathrm{T}}$, где $d^{id}_i \sim U(-1, 1)$, $i = 1, ..., n$.

Шаг 2.4.6. Выполнить нормировку вектора направления $d^{id} = d^{id} / \| d^{id} \|$, где $\| \cdot \|$ – евклидова норма.

Шаг 2.4.7. Сгенерировать вектор, описывающий возможное перемещение бактерии $p^{id} = (p^{id}_1, ..., p^{id}_n)^{\mathrm{s}}$, где $p^{id}_i \sim U(0, p^i_{move})$, $i = 1, ..., n$.

Шаг 2.4.8. Сгенерировать параметр движения бактерии $g^{id} = \hat{\xi}$, где $\xi \sim N(n, (n/3)^2)$, $\hat{\cdot}$ - операция округления.

Шаг 2.4.9. Если $g^{id} = 0$ или $g^{id} > G_{\max}$, то перейти к шагу 2.4.8, в противном случае – к шагу 2.4.10.

Шаг 2.4.10. Сгенерировать способ передвижения $move^{id} \sim U\{1,2,3\}$ и стратегию $str^{id} \sim U\{1,2,3\}$.

Шаг 2.4.11. Положить $id = id + 1$, если $id \le A_{\max}$, то перейти к шагу 2.4.2, в противном случае – к шагу 3.

Шаг 3. Положить $iter = 1$.

Шаг 4. Если $iter > 1$, то перейти к шагу 5, в противном случае – к шагу 6.

Шаг 5. Генерация популяции-потомка.

Шаг 5.1. Отсортировать кортежи $\{m^{id}, w^{id}, \boldsymbol{b}_l^{id}, d^{id}, p^{id}, g^{id}, move^{id}, str^{id}, path^{id} = \{\boldsymbol{path}_j^{id}\}_j = \varnothing, b_f^{id}\}$, $id = 1,..., A_{\max}$ по возрастанию величины $b_f^{id} = \underline{\boldsymbol{f}(\boldsymbol{b}_l^{id})}$.

Шаг 5.2. Создать пул стратегий $pool_{str} = \bigcup_{id=1}^{A_{\max}} str^{id}$ и пул способов перемещений $pool_{move} = \bigcup_{id=1}^{A_{\max}} move^{id}$.

Шаг 5.3. Генерация бактерий случайным образом.

Шаг 5.3.1. Положить $id = 1$.

Шаг 5.3.2. Сгенерировать вектор, описывающий местоположение $m^{id} = (m_1^{id},..., m_n^{id})^{\mathrm{T}}$, где $m_i \sim U(\underline{\boldsymbol{s}_i}, \overline{\boldsymbol{s}_i})$, $i = 1,..., n$.

Шаг 5.3.3. Сгенерировать вектор, описывающий ширину $w^{id} = (w_1^{id},..., w_n^{id})^{\mathrm{T}} = (|v_1| \cdot \omega(\boldsymbol{s}_1),..., |v_n| \cdot \omega(\boldsymbol{s}_n))^{\mathrm{T}}$, где $v_i \sim N(0, 0.25^2)$, $i = 1,..., n$.

Шаг 5.3.4. Используя сгенерированную точку, создать брус $\boldsymbol{b}_l^{id} = \left\langle [m_1^{id} - w_1^{id}/2; m_1^{id} + w_1^{id}/2] \times ... \times [m_n^{id} - w_n^{id}/2; m_n^{id} + w_n^{id}/2] \right\rangle$, где $\langle \cdot \rangle$ – процедура восстановления (см. разд. 1.3.7).

Шаг 5.3.5. Сгенерировать вектор, описывающий направление $d^{id} = (d_1^{id},..., d_n^{id})^{\mathrm{T}}$, где $d_i^{id} \sim U(-1,1)$, $i = 1,..., n$.

Шаг 5.3.6. Выполнить нормировку вектора направления $d^{id} = d^{id} / \| d^{id} \|$, где $\| \cdot \|$ – евклидова норма.

Шаг 5.3.7. Сгенерировать вектор, описывающий возможное перемещение бактерии, $p^{id} = (p_1^{id},..., p_n^{id})^{\mathrm{T}}$, где $p_i^{id} \sim U(0, p_{move}^{i})$, $i = 1,..., n$.

Шаг 5.3.8. Сгенерировать параметр движения бактерии $g^{id} = \hat{\xi}$, где $\xi \sim N(n,(n/3)^2)$, $\hat{}$ - операция округления.

Шаг 5.3.9. Если $g^{id} = 0$ или $g^{id} > G_{\max}$, то перейти к шагу 5.3.8, в противном случае – к шагу 5.3.10.

Шаг 5.3.10. Сгенерировать способ передвижения $move^{id} \sim U\{1,2,3\}$ и стратегию $str^{id} \sim U\{1,2,3\}$.

Шаг 5.3.11. Положить $id = id + 1$, если $id \le 0,3 \cdot A_{\max}$, то перейти к шагу 5.3.2, в противном случае – к шагу 5.4.

Шаг 5.4. Генерация бактерий, двигающихся в направлении, выработанном популяцией-предком.

Шаг 5.4.1. Положить $id = 0,3 \cdot A_{\max} + 1$.

Шаг 5.4.2. Сгенерировать случайное число $i \sim U\{1,2,...,A_{\max}\}$.

Шаг 5.4.3. Положить вектор, описывающий местоположение $m^{id} = \mathrm{mid}(\boldsymbol{path}^{id}_{|path^{id}|})$, где $|\cdot|$ – количество элементов множества.

Шаг 5.4.4. Сгенерировать вектор, описывающий ширину $w^{id} = (w_1^{id},...,w_n^{id})^{\mathrm{T}} = (|v_1| \cdot \omega(\boldsymbol{s}_1),...,|v_n| \cdot \omega(\boldsymbol{s}_n))^{\mathrm{T}}$, где $v_i \sim N(0,0.25^2)$, $i = 1,...,n$.

Шаг 5.4.5. Используя сгенерированную точку, создать брус $\boldsymbol{b}_l^{id} = \left\langle [m_1^{id} - w_1/2; m_1^{id} + w_1/2] \times ... \times [m_n^{id} - w_n/2; m_n^{id} + w_n/2] \right\rangle$, где $\langle\cdot\rangle$ – процедура восстановления.

Шаг 5.4.6. Положить вектор, описывающий направление $d^{id} = \mathrm{mid}(\boldsymbol{path}^{id}_{|path^{id}|}) - \mathrm{mid}(\boldsymbol{path}^{id}_{|path^{id}|-1})$.

Шаг 5.4.7. Выполнить нормировку вектора направления $d^{id} = d^{id} / \| d^{id} \|$, где $\|\cdot\|$ – евклидова норма.

Шаг 5.4.8. Сгенерировать вектор, описывающий возможное перемещение бактерии, $p^{id} = (p_1^{id},..., p_n^{id})^{\mathrm{T}}$, где $p_i^{id} \sim U(0, p_{move}^i)$, $i = 1,...,n$.

Шаг 5.4.9. Сгенерировать параметр движения бактерии $g^{id} = \hat{\xi}$, где $\xi \sim N(n,(n/3)^2)$, $\hat{}$ - операция округления..

Шаг 5.4.10. Если $g^{id} = 0$ или $g^{id} > G_{\max}$, то перейти к шагу 5.4.9, в противном случае – к шагу 5.4.11.

Шаг 5.4.11. Сгенерировать два случайных числа $i \sim U\{1,2,...,|\, pool_{move}\,|\}$ и $j \sim U\{1,2,...,|\, pool_{str}\,|\}$, где $|\cdot|$ – количество элементов множества.

Шаг 5.4.12. В качестве способа передвижения $move^{id}$ и стратегии str^{id} выбрать i-й и j-й элементы множеств $pool_{move}$ и $pool_{str}$ соответственно (которые удаляются из пула).

Шаг 5.4.13. Положить $id = id + 1$, если $id \le 0,6 \cdot A_{\max}$, то перейти к шагу 5.4.2, в противном случае – к шагу 5.5

Шаг 5.5. Генерация бактерий на начальных позициях популяции-предка, но двигающихся в противоположном направлении.

Шаг 5.5.1. Положить $id = 0,6 \cdot A_{\max} + 1$.

Шаг 5.5.2. Сгенерировать случайное число $i \sim U\{1,2,..., A_{\max}\}$.

Шаг 5.5.3. Положить вектор, описывающий местоположение $m^{id} = \mathrm{mid}(\boldsymbol{path}_1^{id})$, где $|\cdot|$ – количество элементов множества.

Шаг 5.5.4. Сгенерировать вектор, описывающий ширину $w^{id} = (w_1^{id},..., w_n^{id})^{\mathrm{T}} = (|\, v_1\,| \cdot \omega(\boldsymbol{s}_1),..., |\, v_n\,| \cdot \omega(\boldsymbol{s}_n))^{\mathrm{T}}$, где $v_i \sim N(0, 0.25^2)$, $i = 1,..., n$.

Шаг 5.5.5. Используя сгенерированную точку, создать брус $\boldsymbol{b}_l^{id} = \left\langle [m_1^{id} - w_1^{id}/2; m_1^{id} + w_1^{id}/2] \times ... \times [m_n^{id} - w_n^{id}/2; m_n^{id} + w_n^{id}/2] \right\rangle$, где $\langle \cdot \rangle$ - процедура восстановления.

Шаг 5.5.6. Положить вектор, описывающий направление $d^{id} = \mathrm{mid}(\boldsymbol{path}_1^{id}) - \mathrm{mid}(\boldsymbol{path}_{path^{id}}^{id})$.

Шаг 5.5.7. Выполнить нормировку вектора направления $d^{id} = d^{id} / \| d^{id} \|$, где $\|\cdot\|$ – евклидова норма.

Шаг 5.5.8. Сгенерировать вектор, описывающий возможное перемещение бактерии, $p^{id} = (p_1^{id},..., p_n^{id})^{\mathrm{T}}$, где $p_i^{id} \sim U(0, p_{move}^i)$, $i = 1,..., n$.

Шаг 5.5.9. Сгенерировать параметр движения бактерии $g^{id} = \hat{\xi}$, где $\xi \sim N(n, (n/3)^2)$, $\hat{\cdot}$ - операция округления..

Шаг 5.5.10. Если $g^{id} = 0$ или $g^{id} > G_{\max}$, то перейти к шагу 5.5.9, в противном случае – к шагу 5.5.11.

Шаг 5.5.11. Сгенерировать два случайных числа $i \sim U\{1,2,...,|\, pool_{move}\,|\}$ и $j \sim U\{1,2,...,|\, pool_{str}\,|\}$, где $|\cdot|$ – количество элементов множества.

Шаг 5.5.12. В качестве способа передвижения $move^{id}$ и стратегии str^{id} выбрать i-й и j-й элементы множеств $pool_{move}$ и $pool_{str}$ соответственно (которые удаляются из пула).

Шаг 5.5.13. Положить $id = id + 1$, если $id \le 0,9 \cdot A_{\max}$, то перейти к шагу 5.5.2, в противном случае – к шагу 5.6

Шаг 5.6. Генерация бактерий, начинающих из текущей лучшей позиции в случайных направлениях.

Шаг 5.6.1. Положить $id = 0,9 \cdot A_{\max} + 1$.

Шаг 5.6.2. Определить вектор, описывающий местоположение, $m^{id} = \mathrm{mid}(\boldsymbol{bp})$.

Шаг 5.6.3. Сгенерировать вектор, описывающий ширину $w^{id} = (w_1^{id}, ..., w_n^{id})^{\mathrm{T}} = (|v_1| \cdot \omega(\boldsymbol{s}_1), ..., |v_n| \cdot \omega(\boldsymbol{s}_n))^{\mathrm{T}}$, где $v_i \sim N(0, 0.25^2)$, $i = 1, ..., n$.

Шаг 5.6.4. Используя сгенерированную точку, создать брус $\boldsymbol{b}_l^{id} = \left\langle [m_1^{id} - w_1^{id}/2; m_1^{id} + w_1^{id}/2] \times ... \times [m_n^{id} - w_n^{id}/2; m_n^{id} + w_n^{id}/2] \right\rangle$, где $\langle \cdot \rangle$ – процедура восстановления (см. разд. 1.3.7).

Шаг 5.6.5. Сгенерировать вектор, описывающий направление $d^{id} = (d_1^{id}, ..., d_n^{id})^{\mathrm{T}}$, где $d_i^{id} \sim U(-1,1)$, $i = 1, ..., n$.

Шаг 5.6.6. Выполнить нормировку вектора направления $d^{id} = d^{id} / \| d^{id} \|$, где $\| \cdot \|$ – евклидова норма.

Шаг 5.6.7. Сгенерировать вектор, описывающий возможное перемещение бактерии, $p^{id} = (p_1^{id}, ..., p_n^{id})^{\mathrm{T}}$, где $p_i^{id} \sim U(0, p_{move}^i)$, $i = 1, ..., n$.

Шаг 5.6.8. Сгенерировать параметр движения бактерии $g^{id} = \hat{\xi}$, где $\xi \sim N(n, (n/3)^2)$, $\hat{\cdot}$ - операция округления.

Шаг 5.6.9. Если $g^{id} = 0$ или $g^{id} > G_{\max}$, то перейти к шагу 5.6.8, в противном случае – к шагу 5.6.10.

Шаг 5.6.10. Сгенерировать два случайных числа $i \sim U\{1, 2, ..., |pool_{move}|\}$ и $j \sim U\{1, 2, ..., |pool_{str}|\}$, где $|\cdot|$ – количество элементов множества.

Шаг 5.6.11. В качестве способа передвижения $move^{id}$ и стратегии str^{id} выбрать i-й и j-й элементы множеств $pool_{move}$ и $pool_{str}$ соответственно (которые удаляются из пула).

Шаг 5.6.12. Положить $id = id + 1$, если $id \le A_{\max}$, то перейти к шагу 5.6.2, в противном случае – к шагу 6

Шаг 6. Симуляция цикла жизнедеятельности.

Шаг 6.1. Положить $id = 1$.

Шаг 6.2. Определить множество, описывающее путь передвижения бактерии $path^{id} = \{\boldsymbol{b}_l^{id}\}$.

Шаг 6.3. Если $\max w^{id} > w_\varepsilon$, то перейти к шагу 6.4, в противном случае – к шагу 6.6.

Шаг 6.4. Выполнить процесс перемещения id-й бактерии со способом перемещения $move^{id}$ и со стратегией str^{id}.

Шаг 6.5. Добавить в множество $path^{id}$ обновленный брус $\boldsymbol{b}_l^{id}$ и перейти к шагу 6.3.

Шаг 6.6. Положить $id = id + 1$, если $id \le A_{\max}$, то перейти к шагу 6.2, в противном случае – к шагу 7.

Шаг 7. Выбор лучшей бактерии.

Шаг 7.1. Выбрать наиболее приспособленную бактерию $id^* = \mathrm{Arg}\min_{id=1,\dots,A_{\max}} b_f^{id}$.

Шаг 7.2. Если $b_f^{id^*} < bv$, то перейти к шагу 7.3, в противном случае – к шагу 8.

Шаг 7.3. Обновить текущее лучшее значение $bv = b_f^{id^*}$ и текущий лучший брус $\boldsymbol{bp} = \boldsymbol{b}_l^{id^*}$.

Шаг 8. Положить $iter = iter + 1$, если $iter > N_{\max}$, то перейти к шагу 9, в противном случае – к шагу 4.

Шаг 9. В качестве ответа вернуть брус $\boldsymbol{bp}$.

Процессы перемещения бактерий

«Жадная» стратегия перемещения id-й бактерии ($str^{id} = 1$). Бактерия перемещается в первую найденную область, которая оказалась лучше, чем текущая.

Шаг 1. Обновление матрицы трансформаций Ω^{id} (см. далее).

Шаг 2. Положить $iter = 1$.

Шаг 3. Сгенерировать точку $\tilde{m}$ в соответствии со способом передвижения $move^{id}$.

Шаг 4. Положить брус $\tilde{\boldsymbol{b}}_l^{id} = \langle [\tilde{m}_1 - w_1/2; \tilde{m}_1 + w_1/2] \times \dots \times [\tilde{m}_n - w_n/2; \tilde{m}_n + w_n/2] \rangle$, где $\langle\cdot\rangle$ - процедура восстановления (см. разд. 1.3.7), w_i – ширина i-й компоненты бруса $\tilde{\boldsymbol{b}}_l^{id}$.

Шаг 5. Положить $\tilde{b}_f = \underline{\boldsymbol{f}(\tilde{\boldsymbol{b}}_l^{id})}$.

Шаг 6. Если $\tilde{b}_f < b_f^{id}$, то перейти к шагу 7, в противном случае – к шагу 8.

Шаг 7. Положить $b_f^{id} = \tilde{b}_f$, $\boldsymbol{b}_l^{id} = \tilde{\boldsymbol{b}}_l^{id}$, $d^{id} = \dfrac{\tilde{m} - m^{id}}{\left\| \tilde{m} - m^{id} \right\|}$, $m^{id} = \tilde{m}$, $w_i^{id} = \max\{\xi_i, 1\} \cdot w_i^{id}$, $i = 1, 2, ..., n$

($\xi_i \sim N(0.8, 0.1^2)$ - случайная величина; если полученное значение $\xi_i \geq 1$ или $\xi_i \leq 0$, то величины генерируется вновь) и закончить работу алгоритма.

Шаг 8. Положить $iter = iter + 1$. Если $iter \leq g^{id}$, то перейти к шагу 3, в противном случае – закончить работу алгоритма.

«Умная» стратегия перемещения id-й бактерии ($str^{id} = 2$). Бактерия перемещается в первую найденную область, которая оказалась лучше, чем текущая.

Шаг 1. Обновление матрицы трансформаций.

Шаг 2. Положить $c_f = b_f^{id}$, $\boldsymbol{c}_l = \boldsymbol{b}_l^{id}$, $n = m^{id}$, $iter = 1$.

Шаг 3. Сгенерировать точку $\tilde{m}$ в соответствии со способом передвижения $move^{id}$.

Шаг 4. Положить брус $\tilde{\boldsymbol{b}}_l^{id} = \left\langle [\tilde{m}_1 - w_1 / 2; \tilde{m}_1 + w_1 / 2] \times ... \times [\tilde{m}_n - w_n / 2; \tilde{m}_n + w_n / 2] \right\rangle$, где $\langle \cdot \rangle$ - процедура восстановления (см. разд. 1.3.7), w_i – ширина i-й компоненты бруса $\tilde{\boldsymbol{b}}_l^{id}$.

Шаг 5. Положить $\tilde{b}_f = \underline{[f]}(\tilde{\boldsymbol{b}}_l^{id})$.

Шаг 6. Если $\tilde{b}_f < c_f$, то перейти к шагу 7, в противном случае – к шагу 8.

Шаг 7. Положить $c_f = \tilde{b}_f$, $\boldsymbol{c}_l = \tilde{\boldsymbol{b}}_l^{id}$, $n = \tilde{m}$.

Шаг 8. Положить $iter = iter + 1$. Если $iter \leq g^{id}$, то перейти к шагу 3, в противном случае – положить $b_f^{id} = c_f$, $\boldsymbol{b}_l^{id} = \boldsymbol{c}_l$, $d^{id} = \dfrac{n - m^{id}}{\left\| n - m^{id} \right\|}$, $m^{id} = n$, $w_i^{id} = \max\{\xi_i, 1\} \cdot w_i^{id}$, $i = 1, 2, ..., n$

($\xi_i \sim N(0.8, 0.1^2)$ - случайная величина; если полученное значение $\xi_i \geq 1$ или $\xi_i \leq 0$, то величины генерируется вновь) и закончить работу алгоритма.

«Цепная» стратегия перемещения id-й бактерии ($str^{id} = 3$). Бактерия производит многократное смещение.

Шаг 1. Обновление матрицы трансформаций.

Шаг 2. Положить $iter = 1$.

Шаг 3. Сгенерировать точку $\tilde{m}$ в соответствии со способом передвижения $move^{id}$.

Шаг 4. Положить брус $\tilde{\boldsymbol{b}}_l^{id} = \left\langle [\tilde{m}_1 - w_1 / 2; \tilde{m}_1 + w_1 / 2] \times ... \times [\tilde{m}_n - w_n / 2; \tilde{m}_n + w_n / 2] \right\rangle$, где $\langle \cdot \rangle$ - процедура восстановления (см. разд. 1.3.7), w_i – ширина i-й компоненты бруса $\tilde{\boldsymbol{b}}_l^{id}$.

Шаг 5. Положить $\tilde{b}_f = \underline{\boldsymbol{f}(\tilde{\boldsymbol{b}}_l^{id})}$.

Шаг 6. Если $\tilde{b}_f < b_f^{id}$, то перейти к шагу 7, в противном случае – к шагу 8.

Шаг 7. Положить $b_f^{id} = \tilde{b}_f$, $\boldsymbol{b}_l^{id} = \tilde{\boldsymbol{b}}_l^{id}$, $d^{id} = \dfrac{\tilde{m} - m^{id}}{\left\| \tilde{m} - m^{id} \right\|}$, $m^{id} = \tilde{m}$, $w_i^{id} = \max\{\xi_i, 1\} \cdot w_i^{id}$, $i = 1, 2, ..., n$

($\xi_i \sim N(0.8, 0.1^2)$ - случайная величина; если полученное значение $\xi_i \geq 1$ или $\xi_i \leq 0$, то величины генерируется вновь).

Шаг 8. Положить $iter = iter + 1$. Если $iter \leq g^{id}$, то перейти к шагу 3, в противном случае – закончить работу алгоритма.

Способы генерации точек

«Пирамидальный» способ передвижения id-й бактерии ($move^{id} = 1$).

Шаг 1. Сгенерировать случайную величину $v_1 \sim U(0, p_1^{id})$.

Шаг 2. Положить величину $r = v_1 / p_1^{id}$.

Шаг 3. Сгенерировать случайные величины $v_i \sim U(-r \cdot p_i^{id}, r \cdot p_i^{id})$, $i = 2, ..., n$.

Шаг 4. Выполнить трансформацию вектора: $v = \Omega^{id} \cdot v + m^{id}$.

«Зональный» способ передвижения id-й бактерии ($move^{id} = 2$).

Шаг 1. Сгенерировать случайные величины $v_i \sim U(-p_i^{id}, p_i^{id})$, $i = 1, ..., n$.

Шаг 2. Выполнить трансформацию вектора: $v = \Omega^{id} \cdot v + m^{id}$.

Способ передвижения *«звезда»* id-й бактерии ($move^{id} = 2$).

Шаг 1. Положить вектор $v = (\underset{n}{0, ..., 0})^T$.

Шаг 2. Сгенерировать случайную величину $i \sim U\{1, 2, ..., n\}$

Шаг 3. Сгенерировать i-ю компоненту вектора $v_i \sim U(-p_i^{id}, p_i^{id})$.

Шаг 4. Выполнить трансформацию вектора: $v = \Omega^{id} \cdot v + m^{id}$.

Замечание. *Алгоритм обновления матрицы трансформаций* Ω^{id}:

Шаг 1. Если $n = 1$, то положить матрицу трансформаций $\Omega^{id} = (d_1^{id})$, в противном случае – положить матрицу трансформаций $\Omega^{id} = E_n$ - единичная матрица $n \times n$, и перейти к шагу 2.

Шаг 2.Положить вектор $v = (\underset{n-1}{1, 0, ..., 0})$.

Шаг 3. Положить вектор $u = (d_1^{id}, \underset{n-1}{0, ..., 0})^T$.

Шаг 4. Положить $i = 2$.

Шаг 5. Положить $u_i = d_i^{id}$.

Шаг 6. Найти угол $\varphi = \arccos \frac{v \cdot u}{\|v\| \cdot \|u\|}$, где $\|\cdot\|$ - евклидова норма (если $\|v\| \cdot \|u\| = 0$, то положить $\varphi = 0$).

Шаг 7. Если $u_i < v_i$, то положить $\varphi = -\varphi$.

Шаг 8. Найти матрицу поворота.

Шаг 8.1. Положить $R = E_n$ - единичная матрица $n \times n$.

Шаг 8.2. Положить $R_{1,1} = \cos\varphi$, $R_{1,i} = -\sin\varphi$, $R_{i,1} = \sin\varphi$, $R_{1,1} = \cos\varphi$.

Шаг 9. Положить $\Omega^{id} = \Omega^{id} \cdot R$.

Шаг 10. Положить $i = i + 1$, если $i \le n$, то перейти к шагу 5, в противном случае – закончить работу алгоритма.

Рекомендации по подбору параметров

Увеличение параметра $N_{\max}$ приводит к одновременному увеличению точности и времени работы алгоритма. Обратные действия приводят к общему ускорению расчета, но за счет снижения точности работы алгоритма.

1.4. ТЕСТИРОВАНИЕ ИНТЕРВАЛЬНЫХ МЕТОДОВ ОПТИМИЗАЦИИ

В данном разделе приведены результаты тестирования разработанных алгоритмов оптимизации на наборе тестовых функций со сложной поверхностью уровня.

Список тестовых функций:

- функция де Йонга:
  - вид функции: $f(x) = \sum_{i=1}^{n} x_i^2$,
  - область поиска: $\underbrace{[-10;10] \times \ldots \times [-10;10]}_{n}$,
  - точка минимума и значение функции: $x^* = \underbrace{(0;\ldots;0)}_{n}^{\mathrm{T}}$, $f(x^*) = 0$,

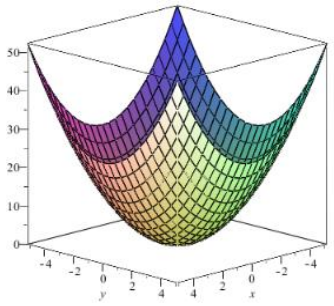


Рис. 1.8. Функция де Йонга при $n = 2$

- функция Растригина:
  - вид функции: $f(x) = 10 \cdot n + \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos(2\pi \cdot x_i))$,
  - область поиска: $\underbrace{[-10;10] \times \ldots \times [-10;10]}_{n}$,
  - точка минимума и значение функции: $x^* = \underbrace{(0;\ldots;0)}_{n}^{\mathrm{T}}$, $f(x^*) = 0$,

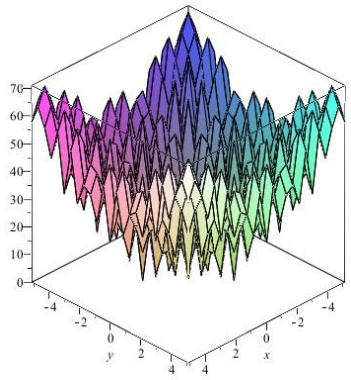


Рис. 1.9. Функция Растригина при $n = 2$

- функция Швефеля:
  - вид функции: $f(x) = -\sum_{i=1}^{n} x_i \cdot \sin(\sqrt{|x_i|})$,
  - область поиска: $\underbrace{[-500;500] \times \ldots \times [-500;500]}_{n}$,
  - точка минимума и значение функции: $x^* = \underbrace{(420,9687;\ldots;420,9687)}_{n}^{\mathrm{T}}$, $f(x^*) = -n \cdot 418,9829$,

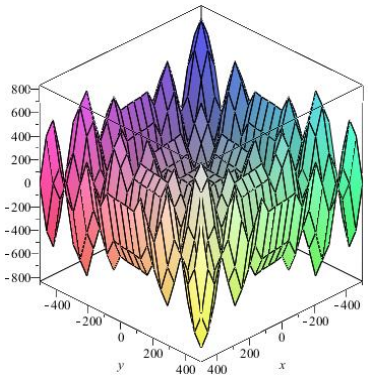


Рис. 1.10. Функция Швефеля при $n = 2$

- функция Изома:
  - вид функции: $f(x) = -\cos(x_1) \cdot \cos(x_2) \cdot \exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$,
  - область поиска: $[-100;100] \times [-100;100]$,

- точка минимума и значение функции: $x^* = (\pi;\pi)^T$, $f(x^*) = -1$,

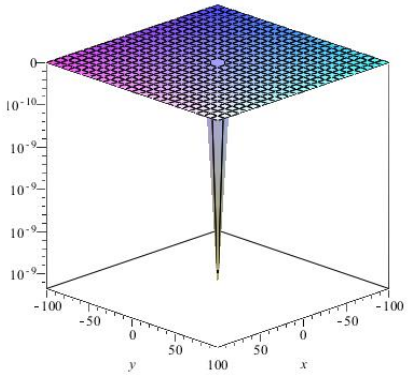


Рис. 1.11. Функция Изома

- функция Экли:
  - вид функции: $f(x) = -20 \cdot \exp(-0,2 \cdot \sqrt{0,5 \cdot (x_1^2 + x_2^2)}) - \exp(0,5 \cdot (\cos(2\pi \cdot x_1) + \cos(2\pi \cdot x_2)) + 20 + e$,
  - область поиска: $[-32,768; 32,768] \times [-32,768; 32,768]$,
  - точка минимума и значение функции: $x^* = (0;0)^T$, $f(x^*) = 0$,

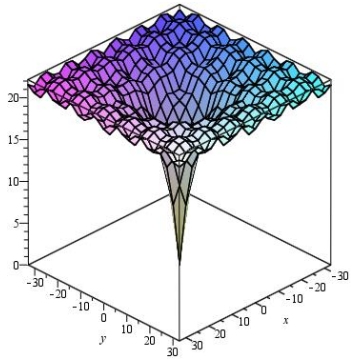


Рис. 1.12. Функция Экли

- функция Биля:
  - вид функции: $f(x) = (1,5 - x_1(1 - x_2))^2 + (2,25 - x_1(1 - x_2^2))^2 + (2,625 - x_1(1 - x_2^3))^2$,
  - область поиска: $[-4,5; 4,5] \times [-4,5; 4,5]$,
  - точка минимума и значение функции: $x^* = (3; 0,5)^T$, $f(x^*) = 0$,

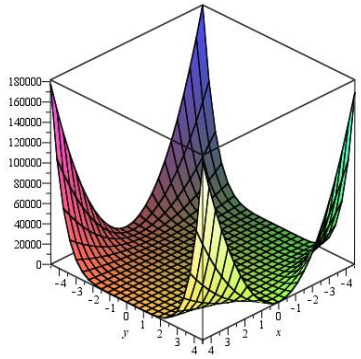


Рис. 1.13. Функция Биля

- функция Розенброка:
  - вид функции: $f(x) = (1 - x_1)^2 + 100 \cdot (x_2 - x_1^2)^2$,
  - область поиска: $[-5;10] \times [-5;10]$,
  - точка минимума и значение функции: $x^* = (1;1)^{\mathrm{T}}$, $f(x^*) = 0$.

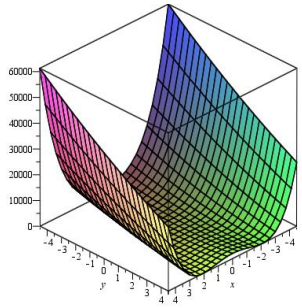


Рис. 1.14. Функция Розенброка

1.4.1. МЕТОД ДИХОТОМИИ ЦЕЛЕВОГО ИНТЕРВАЛА

При решении тестовых задач были выбраны следующие параметры работы алгоритма: $\varepsilon = \zeta = 0,01$.

Табл. 1.1. Результаты работы

Название функции	Найденный брус
Функция де Йонга ($n = 2$)	$[0;0,01] \times [0;0,01]$
Функция Растригина ($n = 2$)	$[0;0,01] \times [0;0,01]$
Функция Швефеля ($n = 2$)	$[420,967;420,974] \times [420,967;420,974]$
Функция Изома	$[3,137;3,143] \times [3,137;3,143]$
Функция Экли	$[-0,008;0] \times [-0,008;0]$
Функция Биля	$[2,953;2,961] \times [0,483;0,492]$
Функция Розенброка	$[0,998;1,005] \times [0,991;0,998]$

1.4.2. МЕТОД ОТСЕЧКИ ВИРТУАЛЬНЫХ ЗНАЧЕНИЙ

При решении тестовых задач были выбраны следующие параметры работы алгоритма: $\delta = 5, \varepsilon = \zeta = 0,01$.

Табл. 1.2. Результаты работы

Название функции	Найденный брус
Функция де Йонга ($n = 2$)	$[0;0,01] \times [0;0,01]$
Функция Растригина ($n = 2$)	$[0;0,01] \times [0;0,01]$
Функция Швефеля ($n = 2$)	$[420,967;420,974] \times [420,967;420,974]$
Функция Изома	$[3,137;3,143] \times [3,137;3,143]$
Функция Экли	$[-0,008;0] \times [-0,008;0]$
Функция Биля	$[2,953;2,961] \times [0,483;0,492]$
Функция Розенброка	$[0,998;1,005] \times [0,991;0,998]$

1.4.3. МЕТОД СТОХАСТИЧЕСКОЙ ОТСЕЧКИ ВИРТУАЛЬНЫХ ЗНАЧЕНИЙ

При решении тестовых задач были выбраны следующие параметры работы алгоритма: $\delta = 5, \varepsilon = \zeta = 0,01$.

Табл. 1.3. Результаты работы

Название функции	Найденный брус
Функция де Йонга ($n = 2$)	$[0;0,01]\times[0;0,01]$
Функция Растригина ($n = 2$)	$[0;0,01]\times[0;0,01]$
Функция Швефеля ($n = 2$)	$[420,967;420,974]\times[420,967;420,974]$
Функция Изома	$[3,137;3,143]\times[3,137;3,143]$
Функция Экли	$[-0,008;0]\times[-0,008;0]$
Функция Биля	$[2,953;2,961]\times[0,483;0,492]$
Функция Розенброка	$[0,998;1,005]\times[0,991;0,998]$

1.4.4. МЕТОД СТОХАСТИЧЕСКИХ ВЫРЫВАНИЙ

При решении тестовых задач были выбраны следующие параметры работы алгоритма: $\varepsilon = \zeta = 0,01, A = 10, W = 5$.

Табл. 1.4. Результаты работы

Название функции	Найденный брус
Функция де Йонга ($n = 2$)	$[0;0,01]\times[0;0,01]$
Функция Растригина ($n = 2$)	$[0;0,01]\times[0;0,01]$
Функция Швефеля ($n = 2$)	$[420,967;420,974]\times[420,967;420,974]$
Функция Изома	$[3,137;3,143]\times[3,137;3,143]$
Функция Экли	$[-0,008;0]\times[-0,008;0]$
Функция Биля	$[2,953;2,961]\times[0,483;0,492]$
Функция Розенброка	$[0,998;1,005]\times[0,991;0,998]$

1.4.5. ОБОБЩЕННЫЙ ИНВЕРСНЫЙ МЕТОД

При решении тестовых задач были выбраны следующие параметры работы алгоритма: оператор проверки OIR, оператор сжатия RPS, $\varepsilon = \zeta = 0,01, A = 10$.

Табл. 1.5, Результаты работы

Название функции	Найденный брус
Функция де Йонга ($n = 2$)	$[0;0,01]\times[0;0,01]$
Функция Растригина ($n = 2$)	$[0;0,01]\times[0;0,01]$
Функция Швефеля ($n = 2$)	$[420,967;420,974]\times[420,967;420,974]$
Функция Изома	$[3,137;3,143]\times[3,137;3,143]$
Функция Экли	$[-0,008;0]\times[-0,008;0]$
Функция Биля	$[2,953;2,961]\times[0,483;0,492]$
Функция Розенброка	$[0,998;1,005]\times[0,991;0,998]$

1.4.6. МЕТОД УСРЕДНЕННЫХ КОНЦОВ ПУТЕЙ

При решении тестовых задач были выбраны следующие параметры работы алгоритма: $\varepsilon = 0,01, v = 0,95, p_{amount} = 10, p_{attempts} = 5, r_{\max} = 5$.

Табл. 1.6. Результаты работы

Название функции	Найденный брус
Функция де Йонга ($n = 2$)	$[-0,005; 0,004] \times [-0,004; 0,005]$
Функция Растригина ($n = 2$)	$[-0,005; 0,004] \times [0,042; 0,046]$
Функция Швефеля ($n = 2$)	$[421,407; 421,416] \times [422,118; 422,127]$
Функция Изома	$[3,136; 3,146] \times [3,136; 3,146]$
Функция Экли	$[-0,005; 0,004] \times [-0,005; 0,005]$
Функция Биля	$[2,993; 3] \times [0,494; 0,503]$
Функция Розенброка	$[0,995; 1,004] \times [0,994; 1,003]$

1.4.7. МЕТОД СТОХАСТИЧЕСКОЙ СЕТКИ

При решении тестовых задач были выбраны следующие параметры работы алгоритма: $\varepsilon = 0,01, v = 0,8, p_{amount} = 5, p_{attempts} = 2$.

Табл. 1.7. Результаты работы

Название функции	Найденный брус
Функция де Йонга ($n = 2$)	$[-0,005; 0,004] \times [-0,003; 0,004]$
Функция Растригина ($n = 2$)	$[-0,004; 0,004] \times [0,005; 0,002]$
Функция Швефеля ($n = 2$)	$[422,315; 422,324] \times [423,590; 423,596]$
Функция Изома	$[3,138; 3,146] \times [3,137; 3,145]$
Функция Экли	$[-0,006; 0,004] \times [-0,004; 0,005]$
Функция Биля	$[2,976; 2,98] \times [0,49; 0,499]$
Функция Розенброка	$[0,964; 0,972] \times [0,934; 0,935]$

1.4.8. МЕТОД ИНТЕРВАЛЬНОГО РАЗБРОСАННОГО ПОИСКА

При решении тестовых задач были выбраны следующие параметры работы алгоритма: $\varepsilon = 0,01, p_{size} = 20, w = 0,3, rs_{size} = 5, b = 0,5, sub_{size} = 2$.

Табл. 1.8. Результаты работы

Название функции	Найденный брус
Функция де Йонга ($n = 2$)	$[-0,005; 0,002] \times [-0,001; 0,006]$
Функция Растригина ($n = 2$)	$[-0,007; 0,002] \times [0; 0,004]$
Функция Швефеля ($n = 2$)	$[410,663; 410,667] \times [420,97; 420,975]$
Функция Изома	$[3,138; 3,145] \times [3,141; 3,15]$
Функция Экли	$[-0,004; 0,006] \times [-0,007; 0,006]$
Функция Биля	$[3,108; 3,112] \times [0,526; 0,53]$
Функция Розенброка	$[1,006; 1,015] \times [1,015; 1,017]$

1.4.9. ИНТЕРВАЛЬНЫЙ ГЕНЕТИЧЕСКИЙ АЛГОРИТМ

При решении тестовых задач были выбраны следующие параметры работы алгоритма: $t_{\max} = 1000, \varepsilon = 0,01, pop_{amount} = 50, pool_{amount} = 20, r = 0,05$, LR кодирование, одноточечные скрещивание и мутация, рулетка, простое усечение.

Табл. 1.9. Результаты работы

Название функции	Найденный брус
Функция де Йонга ($n = 2$)	$[-0,009; 0,001] \times [-0,001; 0,006]$
Функция Растригина ($n = 2$)	$[-0,005; 0,003] \times [0,01; 0,016]$
Функция Швефеля ($n = 2$)	$[410,659; 410,668] \times [420,964; 420,975]$
Функция Изома	$[3,138; 3,146] \times [3,141; 3,15]$
Функция Экли	$[-0,004; 0,005] \times [-0,007; 0,006]$
Функция Биля	$[3,109; 3,115] \times [0,523; 0,53]$
Функция Розенброка	$[1,006; 1,015] \times [1,012; 1,013]$

1.4.10. ИНТЕРВАЛЬНЫЙ МЕТОД ВЗРЫВОВ

При решении тестовых задач были выбраны следующие параметры работы алгоритма: $t^g_{\max} = t^c_{\max} = 500, B_{\max} = 20, r_{\max} = (5;5)^{\mathrm{T}}$.

Табл. 1.10. Результаты работы

Название функции	Найденный брус
Функция де Йонга ($n = 2$)	$[-0,004; 0,002] \times [-0,001; 0,005]$
Функция Растригина ($n = 2$)	$[-0,007; 0,001] \times [0; 0,003]$
Функция Швефеля ($n = 2$)	$[420,95; 420,976] \times [420,96; 420,967]$
Функция Изома	$[3,134; 3,145] \times [3,141; 3,17]$
Функция Экли	$[-0,003; 0,006] \times [-0,003; 0,006]$
Функция Биля	$[3,108; 3,111] \times [0,526; 0,53]$
Функция Розенброка	$[1,006; 1,011] \times [1,011; 1,013]$

1.4.11. АДАПТИВНЫЙ ИНТЕРВАЛЬНЫЙ АЛГОРИТМ

При решении тестовых задач были выбраны следующие параметры работы алгоритма: $N_{\max} = 250, \gamma = 0,95, w_{init} = 0,8$, $N_B = 8, \gamma_B = 0,95, C^+_B = C^-_B = 4, \alpha_B = \beta_B = 0,4$, $r^B = (5;5)^{\mathrm{T}}$, $q^+ = q^- = (0,4; 0,3; 0,2; 0,1)^{\mathrm{T}}$, $N_W = 5, Sp = (10;10)^{\mathrm{T}}$, $\rho_W = 0,2, \gamma_W = 0,9$, $A_{good} = 3$, $A_{bad} = 2$, $N_A = 10, \gamma_A = 0,9, A_{\max} = 10, r^A = (5,5)^{\mathrm{T}}$.

Табл. 1.11. Результаты работы

Название функции	Найденный брус
Функция де Йонга ($n = 2$)	$[-0,004; 0,003] \times [-0,002; 0,004]$
Функция Растригина ($n = 2$)	$[-0,003; 0,003] \times [-0,001; 0,003]$
Функция Швефеля ($n = 2$)	$[420,962; 420,97] \times [420,962; 420,968]$
Функция Изома	$[3,133; 3,143] \times [3,142; 3,17]$
Функция Экли	$[-0,003; 0,006] \times [-0,003; 0,005]$
Функция Биля	$[3,108; 3,109] \times [0,526; 0,531]$
Функция Розенброка	$[1,006; 1,009] \times [1,01; 1,013]$

1.4.12. САМООРГАНИЗУЮЩИЙСЯ ИНТЕРВАЛЬНЫЙ АЛГОРИТМ ИМИТАЦИИ ЭВОЛЮЦИИ КОЛОНИИ БАКТЕРИЙ

При решении тестовых задач были выбраны следующие параметры работы алгоритма: $N_{\max} = 250, w_{\varepsilon} = 0,01$.

Табл. 1.12. Результаты работы

Название функции	Найденный брус
Функция де Йонга ($n = 2$)	$[-0,003; 0,005] \times [-0,003; 0,003]$
Функция Растригина ($n = 2$)	$[-0,005; 0,001] \times [0,001; 0,002]$
Функция Швефеля ($n = 2$)	$[420,965; 420,967] \times [420,961; 420,967]$
Функция Изома	$[3,133; 3,144] \times [3,141; 3,17]$
Функция Экли	$[-0,002; 0,006] \times [-0,003; 0,005]$
Функция Биля	$[3,108; 3,112] \times [0,525; 0,53]$
Функция Розенброка	$[1,005; 1,009] \times [1,010; 1,013]$

1.5. ЗАКЛЮЧЕНИЕ

В первой главе:

1) предложена постановка задачи интервальной ε-минимизации,
2) разработаны интервальные алгоритмы оптимизации на основе инвертера (дихотомии целевого интервала, отсечки виртуальных значений, стохастической отсечки виртуальных значений, стохастических вырываний, обобщенный инверсный метод) и доказаны две теоремы о свойствах решения, полученных с их помощью,
3) разработаны интервальные метаэвристические алгоритмы оптимизации (среднего пути, стохастической сетки, интервального разбросанного поиска, интервальный генетический алгоритм, интервальный метод взрывов, адаптивный интервальный алгоритм, самоорганизующийся интервальный алгоритм имитации эволюции колонии бактерий),

4) приведены замечания и рекомендации относительно подбора параметров работы алгоритмов,
5) разработанные алгоритмы были протестированы при решении типовых тестовых задач.

На основе полученных численных данных можно сделать следующие выводы: несмотря на возможность получения гарантированных результатов, прикладное применение инверсных методов крайне затруднительно. Это объясняется тем, что используемая процедура инвертирования на задачах большой размерности приведет к экспоненциальному росту количества анализируемых брусов (т.н. проклятье размерности), что вызовет переполнение памяти используемого вычислительного устройства. Мета-эвристические интервальные алгоритмы лишены этого недостатка, что позволяет их применять для большего круга задач. Большое количество разработанных алгоритмов объясняется тем, что каждый алгоритм лучше себя проявляет на определенном классе функций. Например, методы усредненных концов путей и стохастической сетки так же лучше работают на задачах малой размерности в связи с реализацией процедуры построения сетки на брусе поиска. Самоорганизующийся интервальный алгоритм имитации эволюции колонии бактерий рекомендуется применять в случае, когда неизвестно почти ничего об оптимизируемой функции, т.к. метод почти не требует параметров. Адаптивный и генетический интервальные алгоритмы рекомендуется применять в обратной ситуации: в связи с тем, что методы обладают большим количеством параметров, они является наиболее гибким из разработанных, что позволяет по максимуму использовать имеющуюся информацию. В связи со всем вышеперечисленным рекомендуется одновременное применение нескольких алгоритмов для получения лучшего результата.

ГЛАВА 2. ИНТЕРВАЛЬНЫЕ АЛГОРИТМЫ СИНТЕЗА ОПТИМАЛЬНЫХ ДИНАМИЧЕСКИХ СИСТЕМ

Во данной главе предложены методы приближенного решения трех задач оптимального управления, в которых применяются разработанные в главе 1 интервальные алгоритмы глобальной условной оптимизации.

2.1. ИНТЕРВАЛЬНЫЕ АЛГОРИТМЫ НАХОЖДЕНИЯ ОПТИМАЛЬНОГО ПРОГРАММНОГО УПРАВЛЕНИЯ НЕЛИНЕЙНЫМИ ДЕТЕРМИНИРОВАННЫМИ ДИНАМИЧЕСКИМИ СИСТЕМАМИ

2.1.1. ПОСТАНОВКА ЗАДАЧИ

Поведение модели объекта управления описывается дифференциальным уравнением

$$\dot{x}(t) = f(t, x(t), u(t)), \tag{2.1}$$

где $t \in \boldsymbol{T} = [t_0; t_1]$ – непрерывное время и соответствующий ему промежуток времени функционирования системы, $x \in \mathbb{R}^n$ – вектор состояния системы, $u \in \boldsymbol{U} \subset \mathbb{R}^q$ – вектор управления; $\boldsymbol{U} = \boldsymbol{U}_1 \times \ldots \times \boldsymbol{U}_q$ – множество допустимых значений управления, представляющее собой брус, $f(t,x,u) = (f_1(t,x,u), \ldots, f_n(t,x,u))^T$ – непрерывно-дифференцируемая вектор-функция.

Начальное состояние задано и равно

$$x(t_0) = x_0 . \tag{2.2}$$

В момент окончания функционирования системы t_1 должны выполняться условия

$$\Gamma_i(t_1, x(t_1)) \in \boldsymbol{G}_i, i = 1, ..., l , \tag{2.3}$$

где $0 \le l \le n+1$, функции $\Gamma_i(t_1, x)$ – непрерывно дифференцируемые; система векторов $\{\partial\Gamma_i(t_1,x)/\partial x_1, \ldots, \partial\Gamma_i(t_1,x)/\partial x_n, \partial\Gamma_i(t_1,x)/\partial t_1\}$ линейно независима $\forall (t_1, x) \in \mathbb{R}^{n+1}$, а $\boldsymbol{G}_i, i = 1, ..., l$ – заданные интервалы.

При управлении используется информация только о времени t (применяется программное управление).

Множество допустимых процессов $D(t_0, x_0)$ определяется как множество троек $d = (t_1, x(\cdot), u(\cdot))$, включающих момент окончания функционирования системы t_1, кусочно-

дифференцируемую траекторию $x(\cdot)$ и кусочно-непрерывное управление $u(\cdot)$, где $u(t) \in \boldsymbol{U}, \forall t \in \boldsymbol{T}$, удовлетворяющих уравнению состояния (2.1), условиям (2.2) и (2.3).

На множестве допустимых процессов $D(t_0, x_0)$ определен функционал качества управления

$$I(x_0, d) = \int_{t_0}^{t_1} f^0(t, x(t), u(t))dt + F(t_1, x(t_1)), \tag{2.4}$$

где $f^0(t, x, u)$, $F(t_1, x)$ – заданные непрерывные функции.

Для учёта конечных условий (2.3) и фазовых ограничений $\{p_j(x(t), u(t), \dot{x}(t), \dot{u}(t)) \in \boldsymbol{P}_j, \forall t \in \boldsymbol{T}\}_{j=1}^{N_p}$ к терминальному члену функционала (2.4) могут быть добавлены штрафные функции, характеризующие степень невыполнения ограничений:

$$\overline{I}(x_0, d) = I(x_0, d) + \sum_{i=1}^{l} R_i^{(1)} \cdot H_i^{(1)} + \sum_{j=1}^{N_p} R_j^{(2)} \cdot H_j^{(2)}, \tag{2.5}$$

где $H_i^{(1)} = h_\infty^0(\Gamma_i(t_1, x(t_1)), \boldsymbol{G}_i)$ и $H_j^{(2)} = \int_{t_0}^{t_1} h_\infty^0(p_j(x(t), u(t), \dot{x}(t), \dot{u}(t)), \boldsymbol{P}_j)dt$ – величины, характеризующие степень невыполнения ограничений (см. разд. 1.1, зам. 2), $R_i^{(1)}, R_j^{(2)}$ – параметры штрафов, $i = 1, ..., l$, $j = 1, ..., N_p$. Применение штрафных функций позволяет не учитывать условия (2.3) в определении множества $D(t_0, x_0)$.

Требуется найти тройку $d^* = (t_1^*, x^*(\cdot), u^*(\cdot)) \in D(t_0, x_0)$ на которой достигается минимальное значение функционала (2.5) на множестве допустимых процессов.

Для решения поставленной задачи предлагается задать структуру управления в параметрическим виде, учитывающем ограничения на управление, и, таким образом, свести задачу к конечномерной. Получаемое в результате управление является субоптимальным.

2.1.2. СТРАТЕГИЯ ПОИСКА УПРАВЛЕНИЯ

Будем предполагать, что компоненты управления $u(t) = (u_1(t), \ldots, u_q(t))^T$ ищутся в одной из следующих интервальных форм:

- кусочно-постоянное интервальное управление; для управления такого типа необходимо задать значения функций $u_i(t), i = \overline{i, q}$ в N моментах времени $\tau_j = t_0 + \frac{t_1 - t_0}{N} \cdot j$, $j = \overline{0, N-1}$, т.е. управлению можно однозначно сопоставить интервальный вектор

$\boldsymbol{a} = \underbrace{\boldsymbol{u}_1(\tau_0)\times\ldots\times\boldsymbol{u}_q(\tau_0)}_{\boldsymbol{u}(\tau_0)}\times\ldots\times\underbrace{\boldsymbol{u}_1(\tau_{N-1})\times\ldots\times\boldsymbol{u}_q(\tau_{N-1})}_{\boldsymbol{u}(\tau_{N-1})}$; соответствующее интервальное управление будет находиться по формуле $\boldsymbol{u}_i(t) = \boldsymbol{u}_i(\tau_j) \subseteq \boldsymbol{U}_i, \tau_j \le t < \tau_{j+1}, i = \overline{1,q},\ j = \overline{0,N-1}$;

- кусочно-линейное интервальное управление; для управления такого типа необходимо задать дополнительное значение управления на последнем временном интервале, поэтому интервальный вектор, который ставится в соответствие управлению, имеет вид $\boldsymbol{a} = \underbrace{\boldsymbol{u}_1(\tau_0)\times\ldots\times\boldsymbol{u}_q(\tau_0)}_{\boldsymbol{u}(\tau_0)}\times\ldots\times\underbrace{\boldsymbol{u}_1(\tau_N)\times\ldots\times\boldsymbol{u}_q(\tau_N)}_{\boldsymbol{u}(\tau_N)}$; соответствующее интервальное управление будет находиться по формуле $\boldsymbol{u}_i(t) = \dfrac{\tau_{j+1}-t}{\tau_{j+1}-\tau_j}\cdot\boldsymbol{u}_i(\tau_j) + \dfrac{t-\tau_j}{\tau_{j+1}-\tau_j}\cdot\boldsymbol{u}_i(\tau_{j+1}) \subseteq [U_i], \tau_j \le t < \tau_{j+1}, i = \overline{1,q},\ j = \overline{0,N-1}$;
- в виде разложения по системам ортонормированных базисных функций (например, полиномов Лежандра $P_n^L(x) = (2^n\ n!)^{-1} d^n\left(x^2-1\right)^n / dx^n$ [65]); для управления такого типа необходимо задать коэффициенты разложения функций $\boldsymbol{u}_i(t) = \mathrm{sat}(\sum_{j=0}^{A}\boldsymbol{a}_i^j\cdot\varphi^j(t), \boldsymbol{U}_i), i = \overline{1,q}$, где $\{\varphi^j(t)\}_{j=0}^{A}$ - система ортонормированных базисных функций, A - масштаб усечения, определяющий количество базисных функций; таким образом, интервальный вектор, который ставится в соответствие управлению, имеет вид $\boldsymbol{a} = \underbrace{\boldsymbol{a}_1^0\times\ldots\times\boldsymbol{a}_1^A}_{A+1}\times\ldots\times\underbrace{\boldsymbol{a}_q^0\times\ldots\times\boldsymbol{a}_q^A}_{A+1}$; соответствующее интервальное управление будет находиться по формуле $\boldsymbol{u}_i(t) = \mathrm{sat}(\underbrace{\sum_{j=0}^{A}\boldsymbol{a}_i^j\cdot\varphi^j(t)}_{\boldsymbol{\varphi}_a(t)}, \boldsymbol{U}_i), i = \overline{1,q}$, где $\mathrm{sat}(\boldsymbol{\varphi}_a(t), \boldsymbol{U}_i) = \begin{cases}[\underline{\boldsymbol{U}_i}; \underline{\boldsymbol{U}_i}], \overline{\boldsymbol{\varphi}_a(t)} < \underline{\boldsymbol{U}_i},\\ [\overline{\boldsymbol{U}_i}; \overline{\boldsymbol{U}_i}], \underline{\boldsymbol{\varphi}_a(t)} > \overline{\boldsymbol{U}_i},\\ \boldsymbol{\varphi}_a(t)\cap\boldsymbol{U}_i\end{cases}$ - функция насыщения.

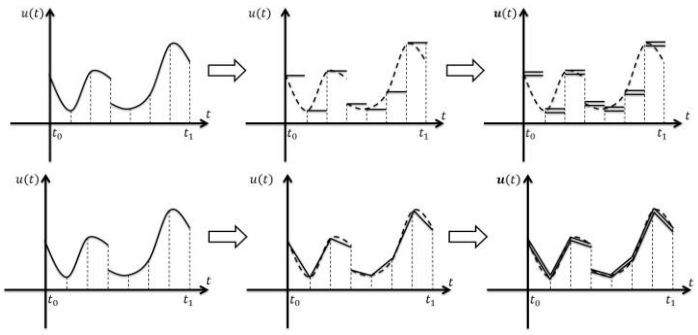


Рис. 2.1. Замена кусочно-непрерывного управления интервальным кусочно-непрерывным и кусочно-линейным

В результате поставленная в разделе 2.1.1 задача сводится к проблеме нахождения наилучшего интервального вектора $\boldsymbol{a}$, минимизирующего функционал (2.5).

2.1.3. АЛГОРИТМ ПОИСКА УПРАВЛЕНИЯ

Для решения задачи поиска наилучшего интервального вектора $\boldsymbol{a}$ и, как следствие, соответствующего ему управления применяются интервальные методы оптимизации, разработанные в главе 1. Вследствие того что данные алгоритмы используются для функций, требуется определить правило, ставящее в соответствие интервальному вектору коэффициентов $\boldsymbol{a}$ значение функционала (2.5). Зададим правило следующим образом:

1) по вектору $\boldsymbol{a}$ в соответствии с выбранным типом представления найти вид управления $\boldsymbol{u}(t)$;
2) найти решение $\boldsymbol{x}(t)$ уравнения модели (2.1) с управлением $\boldsymbol{u}(t)$, начальным состоянием (2.2);
3) вычислить интервальное значение функционала (2.5), величина которого ставится в соответствие вектору $\boldsymbol{a}$.

Таким образом, интервальному вектору $\boldsymbol{a}$ соответствует интервальное управление, используемое в процессе интегрирования уравнений модели объекта (2.1) с начальными условиями (2.2). При подсчете правых частей уравнения (2.1) применяются правила интервальной арифметики [97, 105] (см. приложение). В результате находится интервальное значение критерия (2.5).

2.2. ИНТЕРВАЛЬНЫЕ АЛГОРИТМЫ НАХОЖДЕНИЯ ОПТИМАЛЬНОГО УПРАВЛЕНИЯ С НЕПОЛНОЙ ОБРАТНОЙ СВЯЗЬЮ НЕЛИНЕЙНЫМИ ДЕТЕРМИНИРОВАННЫМИ ДИНАМИЧЕСКИМИ СИСТЕМАМИ

2.2.1. ПОСТАНОВКА ЗАДАЧИ

Поведение модели объекта управления описывается дифференциальным уравнением

$$\dot{x}(t) = f(t, x(t), u(t)), \tag{2.6}$$

где $t \in \boldsymbol{T} = [t_0; t_1]$ – непрерывное время, $x \in \mathbb{R}^n$ – вектор состояния системы, $u \in \boldsymbol{U} \subset \mathbb{R}^q$ – вектор управления; $\boldsymbol{U} = \boldsymbol{U}_1 \times \ldots \times \boldsymbol{U}_q$ – множество допустимых значений управления, представляющее собой брус, $f(t,x,u) = (f_1(t,x,u), \ldots, f_n(t,x,u))^T$ – непрерывно-дифференцируемая вектор-функция.

Возможные начальные состояния системы (2.6) заданы брусом $\boldsymbol{\Omega}$:

$$x(t_0) = x_0 \in \boldsymbol{\Omega} \subset \mathbb{R}^n . \qquad (2.7)$$

Множество $\boldsymbol{\Omega} = \boldsymbol{\omega}_1 \times \ldots \times \boldsymbol{\omega}_n$ характеризует неопределенность задания начальных условий.

В момент окончания функционирования системы t_1 должны выполняться условия

$$\Gamma_i(t_1, x(t_1)) \in \boldsymbol{G}_i, i = 1,...,l , \qquad (2.8)$$

где $0 \le l \le n+1$, функции $\Gamma_i(t_1, x)$ – непрерывно дифференцируемые; система векторов $\{\partial\Gamma_i(t_1, x)/\partial x_1, \ldots, \partial\Gamma_i(t_1, x)/\partial x_n, \partial\Gamma_i(t_1, x)/\partial t_1\}$ линейно независима $\forall (t_1, x) \in \mathbb{R}^{n+1}$, а $\boldsymbol{G}_i, i = 1,...,l$ – заданные интервалы.

Предполагается, что при управлении используется информация о времени t и о части координат вектора состояния x (предполагается, что это первые m координат). Таким образом, о компонентах вектора $x^1 = (x_1, \ldots, x_m)^T \in \mathbb{R}^m$ известна текущая информация, а о компонентах вектора $x^2 = (x_{m+1}, \ldots x_n)^T \in \mathbb{R}^{n-m}$ она отсутствует, при этом $x = (x^1, x^2) \in \mathbb{R}^n$, $0 \le m \le n$. Если $m = 0$, информация о векторе состояния отсутствует, а если $m = n$, то имеется полная информация о векторе состояния.

Управление, применяемое в каждый момент времени t, имеет вид управления с неполной обратной связью: $u(t) = \boldsymbol{u}(t, x^1(t))$. Если $m = 0$, то система управления будет разомкнутой по состоянию, а управление – программным, а если $m = n$ - замкнутой системой с полной обратной связью.

Множество допустимых управлений U образуют такие функции $\boldsymbol{u}(t, x^1)$, что $\forall t \in \boldsymbol{T} : u(t) = \boldsymbol{u}(t, x^1(t)) \in \boldsymbol{U}$, а функция $f(t, x, \boldsymbol{u}(t, x^1))$ такова, что решение уравнения (2.6) существует и единственно.

Множество допустимых процессов $D(t_0, x_0)$ определяется как множество троек $d = (t_1, x(\cdot), u(\cdot))$, включающих момент окончания функционирования системы t_1, кусочно-дифференцируемую траекторию $x(\cdot)$ и кусочно-непрерывное управление $u(\cdot)$, где $u(t) \in \boldsymbol{U}, \forall t \in \boldsymbol{T}$, удовлетворяющих уравнению состояния (2.6), условиям (2.7) и (2.8).

На множестве допустимых процессов $D(t_0, x_0)$ определен функционал качества управления

$$I(x_0, d) = \int_{t_0}^{t_1} f^0(t, x(t), u(t))dt + F(t_1, x(t_1)) , \qquad (2.9)$$

где $f^0(t,x,u)$, $F(t_1,x)$ – заданные непрерывные функции. Для учёта конечных условий (2.8) к терминальному члену функционала (2.9) могут быть добавлены штрафные функции, характеризующие степень их невыполнения (см. разд. 2.1.1):

$$\bar{I}(x_0,d) = I(x_0,d) + \sum_{i=1}^{l} R_i^{(1)} \cdot H_i^{(1)} + \sum_{j=1}^{N_p} R_j^{(2)} \cdot H_j^{(2)}. \tag{2.10}$$

Каждому допустимому управлению $\boldsymbol{u}(t,x^1) \in U$ и брусу $\boldsymbol{\Omega}$ поставим в соответствие пучок (ансамбль) траекторий $x(t,\boldsymbol{u}(t,x^1(t)),x_0)$:

$$X(t,\boldsymbol{u}(t,x^1)) = \bigcup_{x_0 \in \boldsymbol{\Omega}} x(t,\boldsymbol{u}(t,x^1(t)),x_0), \tag{2.11}$$

т.е. объединение решений уравнения (2.6) по всем возможным начальным состояниям (2.7). Пучок траекторий порождается брусом $\boldsymbol{\Omega}$ и управлением $\boldsymbol{u}(t,x^1) \in U$.

Качество управления пучком траекторий предлагается оценивать величиной функционала

$$J[\boldsymbol{u}(t,x^1)] = \frac{\int_{[\boldsymbol{\Omega}]} \bar{I}(x_0,d)dx_0}{\operatorname{mes} \boldsymbol{\Omega}}, \tag{2.12}$$

где $\operatorname{mes} \boldsymbol{\Omega}$ – мера множества $\boldsymbol{\Omega}$.

Требуется найти такое управление $\boldsymbol{u}^*(t,x^1) \in U$, что

$$\boldsymbol{u}^*(t,x^1) = \operatorname{Arg} \min_{\boldsymbol{u}(t,x^1) \in U} J[\boldsymbol{u}(t,x^1)]. \tag{2.13}$$

Искомое управление называется оптимальным в среднем. Для решения поставленной задачи (2.13) предлагается задать структуру управления в параметрическом виде и, таким образом, свести задачу к конечномерной. Получаемое в результате управление является субоптимальным.

2.2.2. СТРАТЕГИЯ ПОИСКА УПРАВЛЕНИЯ

Будем предполагать, что:

- множество начальных состояний $\boldsymbol{\Omega}$ представляет собой брус $\boldsymbol{\Omega} = \boldsymbol{\omega}_1 \times \ldots \times \boldsymbol{\omega}_n$; каждый интервал $\boldsymbol{\omega}_i, i = 1,\ldots,n$ с помощью шага $\Delta_{\omega_i} \geq 0$ (при $\Delta_{\omega_i} = 0$ соответствующая компонента не разбивается, поскольку левая и правая границы интервалов совпадают, а интервал задает точку) разбивается на N_i непересекающихся интервалов, а брус $\boldsymbol{\Omega}$ делится на $N = N_1 \cdot \ldots \cdot N_n$ элементарных непересекающихся подмножеств $\boldsymbol{\Omega}_j$, $j = \overline{1,N}$; в каждом подмножестве $\boldsymbol{\Omega}_j$ задается начальное состояние x_0^j (центр $\boldsymbol{\Omega}_j$);

- компоненты управления $\boldsymbol{u}(t,x^1)=(\boldsymbol{u}_1(t,x^1),\ldots,\boldsymbol{u}_q(t,x^1))^T$ ищутся в виде

$$\boldsymbol{u}_i(t,x^1)=\mathrm{sat}\{g_i(t,x^1),\boldsymbol{U}_i\},i=\overline{1,q}, \tag{2.14}$$

где $\mathrm{sat}(g_i(t,x^1(t)),\boldsymbol{U}_i)=\begin{cases}\underline{\boldsymbol{U}_i},g_i(t,x^1(t))<\underline{\boldsymbol{U}_i},\\ \overline{\boldsymbol{U}_i},g_i(t,x^1(t))>\overline{\boldsymbol{U}_i},\\ g_i(t,x^1(t)),g_i(t,x^1(t))\in\boldsymbol{U}_i,\end{cases}$ - функция насыщения;

- функции $g_i(t,x^1)$ предлагается искать в виде:

$$g_i(t,x^1)=\sum_{k_0=0}^{A_0}\sum_{k_1=0}^{A_1}\ldots\sum_{k_m=0}^{A_m}a^i_{k_0,k_1,\ldots k_m}\cdot\varphi^0_{k_0}(\tilde{t})\cdot\varphi^1_{k_1}(\tilde{x}_1)\cdot\ldots\varphi^m_{k_m}(\tilde{x}_m), \tag{2.15}$$

где $a^i_{k_0,k_1,\ldots,k_m},i=\overline{1,q},k_0=\overline{0,A_0},\ldots,k_m=\overline{0,A_m}$ – неизвестные коэффициенты, $A_j,j=\overline{0,m}$ – масштабы усечения, $\{\varphi^j_{k_j}(\cdot)\}_{k_j=0}^{A_j},j=\overline{0,m}$ – системы базисных функций по переменным, которые используются в управлении, $\tilde{t}=\mathrm{proj}^{\boldsymbol{e}_0}_{\boldsymbol{d}_0}(t),\tilde{x}_1=\mathrm{proj}^{\boldsymbol{e}_1}_{\boldsymbol{d}_1}(x_1),\ldots,\tilde{x}_m=\mathrm{proj}^{\boldsymbol{e}_m}_{\boldsymbol{d}_m}(x_m)$ – проекции значений времени и координат вектора состояния, используемых в управлении, на области определения соответствующих базисных функций, $\boldsymbol{e}_j,j=\overline{0,m}$ – интервалы оценки возможных значений переменной ($\omega(\boldsymbol{e}_j)\neq\infty,j=\overline{0,m}$), $\boldsymbol{d}_j,j=\overline{0,m}$ – области определения используемой системы базисных функций, $\mathrm{proj}^{\boldsymbol{e}}_{\boldsymbol{d}}(x)=\begin{cases}(\omega(\boldsymbol{d})/\omega(\boldsymbol{e}))\cdot(x-\underline{\boldsymbol{e}})+\underline{\boldsymbol{d}},&\omega(\boldsymbol{d})\neq\infty,\\ (x-\underline{\boldsymbol{e}})+\underline{\boldsymbol{d}},&\underline{\boldsymbol{d}}\neq-\infty,\\ x,&\boldsymbol{d}=]-\infty;+\infty[\end{cases}$ – функции проекции величины x на область определения системы базисных функций.

Стратегия поиска управления заключается в переходе от задачи (2.13) к конечномерной задаче поиска минимума приближенного значения функционала (2.13) с помощью подбора коэффициентов, образующих полиномы в (2.15). Для формализации задачи предлагается использовать вектор, являющийся гиперстолбцовой матрицей [66]:

$$a=\begin{pmatrix}\begin{pmatrix}\begin{pmatrix}a^1_{0,0,\ldots,0}\\ a^1_{1,0,\ldots,0}\\ \vdots\\ a^1_{A_0,0,\ldots,0}\end{pmatrix}^{\mathrm{T}}\begin{pmatrix}a^1_{0,1,\ldots,0}\\ a^1_{1,1,\ldots,0}\\ \vdots\\ a^1_{A_0,1,\ldots,0}\end{pmatrix}^{\mathrm{T}}\begin{pmatrix}a^1_{0,2,\ldots,0}\\ a^1_{1,2,\ldots,0}\\ \vdots\\ a^1_{A_0,2,\ldots,0}\end{pmatrix}^{\mathrm{T}}\cdots\begin{pmatrix}a^1_{0,A_1,\ldots,A_m}\\ a^1_{1,A_1,\ldots,A_m}\\ \vdots\\ a^1_{A_0,A_1,\ldots,A_m}\end{pmatrix}^{\mathrm{T}}\end{pmatrix}^{\mathrm{T}}\\ \vdots\\ \begin{pmatrix}\begin{pmatrix}a^q_{0,0,\ldots,0}\\ a^q_{1,0,\ldots,0}\\ \vdots\\ a^q_{A_0,0,\ldots,0}\end{pmatrix}^{\mathrm{T}}\begin{pmatrix}a^q_{0,1,\ldots,0}\\ a^q_{1,1,\ldots,0}\\ \vdots\\ a^q_{A_0,1,\ldots,0}\end{pmatrix}^{\mathrm{T}}\begin{pmatrix}a^q_{0,2,\ldots,0}\\ a^q_{1,2,\ldots,0}\\ \vdots\\ a^q_{A_0,2,\ldots,0}\end{pmatrix}^{\mathrm{T}}\cdots\begin{pmatrix}a^q_{0,A_1,\ldots,A_m}\\ a^q_{1,A_1,\ldots,A_m}\\ \vdots\\ a^q_{A_0,A_1,\ldots,A_m}\end{pmatrix}^{\mathrm{T}}\end{pmatrix}^{\mathrm{T}}\end{pmatrix} \tag{2.16}$$

Соответствующая интервальная гиперстолбцовая матрица:

$$\boldsymbol{a}=\begin{pmatrix}\begin{pmatrix}\begin{pmatrix}\boldsymbol{a}^1_{0,0,\ldots,0}\\ \boldsymbol{a}^1_{1,0,\ldots,0}\\ \vdots\\ \boldsymbol{a}^1_{A_0,0,\ldots,0}\end{pmatrix}^{\mathrm{T}}\begin{pmatrix}\boldsymbol{a}^1_{0,1,\ldots,0}\\ \boldsymbol{a}^1_{1,1,\ldots,0}\\ \vdots\\ \boldsymbol{a}^1_{A_0,1,\ldots,0}\end{pmatrix}^{\mathrm{T}}\begin{pmatrix}\boldsymbol{a}^1_{0,2,\ldots,0}\\ \boldsymbol{a}^1_{1,2,\ldots,0}\\ \vdots\\ \boldsymbol{a}^1_{A_0,2,\ldots,0}\end{pmatrix}^{\mathrm{T}}\cdots\begin{pmatrix}\boldsymbol{a}^1_{0,A_1,\ldots,A_m}\\ \boldsymbol{a}^1_{1,A_1,\ldots,A_m}\\ \vdots\\ \boldsymbol{a}^1_{A_0,A_1,\ldots,A_m}\end{pmatrix}^{\mathrm{T}}\end{pmatrix}^{\mathrm{T}}\\ \vdots\\ \begin{pmatrix}\begin{pmatrix}\boldsymbol{a}^q_{0,0,\ldots,0}\\ \boldsymbol{a}^q_{1,0,\ldots,0}\\ \vdots\\ \boldsymbol{a}^q_{A_0,0,\ldots,0}\end{pmatrix}^{\mathrm{T}}\begin{pmatrix}\boldsymbol{a}^q_{0,1,\ldots,0}\\ \boldsymbol{a}^q_{1,1,\ldots,0}\\ \vdots\\ \boldsymbol{a}^q_{A_0,1,\ldots,0}\end{pmatrix}^{\mathrm{T}}\begin{pmatrix}\boldsymbol{a}^q_{0,2,\ldots,0}\\ \boldsymbol{a}^q_{1,2,\ldots,0}\\ \vdots\\ \boldsymbol{a}^q_{A_0,2,\ldots,0}\end{pmatrix}^{\mathrm{T}}\cdots\begin{pmatrix}\boldsymbol{a}^q_{0,A_1,\ldots,A_m}\\ \boldsymbol{a}^q_{1,A_1,\ldots,A_m}\\ \vdots\\ \boldsymbol{a}^q_{A_0,A_1,\ldots,A_m}\end{pmatrix}^{\mathrm{T}}\end{pmatrix}^{\mathrm{T}}\end{pmatrix}, \tag{2.17}$$

Таким образом, поставленная в разделе 2.2.1 задача сводится к проблеме нахождения наилучшего интервального гиперстолбцового вектора $\boldsymbol{a}$, минимизирующего функционал (2.12).

2.2.3. АЛГОРИТМ ПОИСКА УПРАВЛЕНИЯ

Для решения задачи поиска наилучшего интервального вектора $\boldsymbol{a}$ и, как следствие, управления $\boldsymbol{u}(t,x^1)=(\boldsymbol{u}_1(t,x^1),\ldots,\boldsymbol{u}_q(t,x^1))^T$ далее применяются интервальные методы оптимизации, разработанные в главе 1. Вследствие того что данные алгоритмы применяются для оптимизации функций, требуется определить правило, ставящее в соответствие вектору коэффициентов $\boldsymbol{a}$ значение функционала (2.12). Зададим правило следующим образом:

1) на каждом элементарном множестве $\boldsymbol{\Omega}_j$ задать начальное состояние $x_0^j=\mathrm{mid}(\boldsymbol{\Omega}_j),\ j=\overline{1,N}$ – среднюю точку бруса;
2) по вектору $\boldsymbol{a}$ вида (2.17), размер которого определяется заданными масштабами усечения $A_j,\ j=\overline{0,m}$, найти соответствующее интервальное управление $\boldsymbol{u}(t,x^1)$;
3) найти решение $\boldsymbol{x}^j(t)$ уравнения модели (2.6) с управлением $\boldsymbol{u}(t,x^1)$ и начальным состоянием $x_0^j,\ j=\overline{1,N}$ на промежутке времени $[t_o;t_1^j]$, правый конец которого удовлетворяет (2.8);
4) вычислить интервальное значение функционала (2.9): $I(x_0^j,d^j)$, где $d^j=(t_1^j,\boldsymbol{x}^j(t),\boldsymbol{u}(t)=\boldsymbol{u}(t,\boldsymbol{x}^1(t))),\ j=\overline{1,N}$;
5) после нахождения функционала (2.9) для всех начальных состояний x_0^j из элементарных множеств $\boldsymbol{\Omega}_j$ вычислить критерий

$$J[\boldsymbol{u}(t,x^1)]=\frac{\sum_{j=1}^{N}\bar{I}(x_0^j,d^j)\cdot \operatorname{mes}\boldsymbol{\Omega}_j}{\operatorname{mes}\boldsymbol{\Omega}}=\sum_{j=1}^{N}\bar{I}(x_0^j,d^j)/N\,, \tag{2.18}$$

где $\operatorname{mes}\boldsymbol{\Omega}_j=\prod_{i=1}^{n}(\Delta_{\omega_i}+(1-\operatorname{sign}(\Delta_{\omega_i}))$, $\operatorname{mes}\boldsymbol{\Omega}=\prod_{i=1}^{n}((\overline{\boldsymbol{\omega}_i}-\underline{\boldsymbol{\omega}_i})+(1-\operatorname{sign}((\overline{\boldsymbol{\omega}_i}-\underline{\boldsymbol{\omega}_i})))$. Если по какой-либо координате с номером i^* верхняя и нижняя границы равны: $\overline{\boldsymbol{\omega}_{i^*}}=\underline{\boldsymbol{\omega}_{i^*}}$, т.е. начальное условие задано обычным числом, тогда $(\underbrace{(\overline{\boldsymbol{\omega}_i}-\underline{\boldsymbol{\omega}_i})}_{0}+(1-\underbrace{\operatorname{sign}((\overline{\boldsymbol{\omega}_i}-\underline{\boldsymbol{\omega}_i}))}_{0})=1$, из чего следует, что $\operatorname{mes}\boldsymbol{\Omega}=\prod_{i=i^*-1}^{n}((\overline{\boldsymbol{\omega}_i}-\underline{\boldsymbol{\omega}_i})+(1-\operatorname{sign}((\overline{\boldsymbol{\omega}_i}-\underline{\boldsymbol{\omega}_i})))\cdot 1\cdot\prod_{i=i^*+1}^{n}((\overline{\boldsymbol{\omega}_i}-\underline{\boldsymbol{\omega}_i})+(1-\operatorname{sign}((\overline{\boldsymbol{\omega}_i}-\underline{\boldsymbol{\omega}_i})))$.

Интервальная величина $J[\boldsymbol{u}(t,x^1)]$ ставится в соответствие вектору $\boldsymbol{a}$ и обозначается как $\tilde{J}(\boldsymbol{a})$. В качестве приближенного решения задачи выбирается наилучший вектор $\boldsymbol{a}^*$ и соответствующее ему управление $\boldsymbol{u}^*(t,x^1)$. Для нахождения вектора $\boldsymbol{a}^*$ предлагается использовать интервальные алгоритмы оптимизации, разработанные в главе 1.

2.3. ИНТЕРВАЛЬНЫЕ АЛГОРИТМЫ НАХОЖДЕНИЯ ОПТИМАЛЬНОГО УПРАВЛЕНИЯ ПО ВЫХОДУ НЕЛИНЕЙНЫМИ ДЕТЕРМИНИРОВАННЫМИ ДИНАМИЧЕСКИМИ СИСТЕМАМИ ПРИ НЕОПРЕДЕЛЕННОСТИ В ПАРАМЕТРАХ МОДЕЛИ ОБЪЕКТА УПРАВЛЕНИЯ И МОДЕЛИ ИЗМЕРЕНИЙ

2.3.1. ПОСТАНОВКА ЗАДАЧИ

Поведение модели объекта управления описывается дифференциальным уравнением

$$\dot{x}(t)=f(t,x(t),u(t),p^1)\,, \tag{2.19}$$

где $t\in\boldsymbol{T}=[t_0;t_1]$ – непрерывное время, $x\in\mathbb{R}^n$ – вектор состояния системы, $u\in\boldsymbol{U}\subset\mathbb{R}^q$ – вектор управления; $\boldsymbol{U}=\boldsymbol{U}_1\times\ldots\times\boldsymbol{U}_q$ – множество допустимых значений управления, представляющее собой брус, $p^1\in\mathbb{R}^{s_1}$ – вектор параметров объекта, $f(t,x,u,p)=(f_1(t,x,u,p),\ldots,f_n(t,x,u,p))^T$ – непрерывно-дифференцируемая вектор-функция.

Возможные начальные состояния и значения параметров системы (2.19) заданы брусами $\boldsymbol{\Omega}$ и $\boldsymbol{p}^1$ соответственно:

$$x(t_0)=x_0\in\boldsymbol{\Omega}\subset\mathbb{R}^n,\ p\in\boldsymbol{p}^1\subset\mathbb{R}^{s_1}\,. \tag{2.20}$$

Множество $\boldsymbol{\Omega} = \boldsymbol{\omega}_1 \times \ldots \times \boldsymbol{\omega}_n$ характеризует неопределенность задания начальных условий, а множество $\boldsymbol{p}^1 = \boldsymbol{p}_1^1 \times \ldots \times \boldsymbol{p}_{s_1}^1$ – неопределенность параметров модели объекта управления.

В момент окончания функционирования системы t_1 должны выполняться условия

$$\Gamma_i(t_1, x(t_1)) \in \boldsymbol{G}_i, i = 1, ..., l, \tag{2.21}$$

где $0 \le l \le n+1$, функции $\Gamma_i(t_1, x)$ – непрерывно дифференцируемые; система векторов $\{\partial\Gamma_i(t_1, x)/\partial x_1, \ldots, \partial\Gamma_i(t_1, x)/\partial x_n, \partial\Gamma_i(t_1, x)/\partial t_1\}$ линейно независима $\forall (t_1, x) \in \mathbb{R}^{n+1}$, а $\boldsymbol{G}_i, i = 1, ..., l$ – заданные интервалы.

Предполагается, что модель измерений описывается уравнением выхода

$$z = h(t, x, p^2), \tag{2.22}$$

где $z \in \mathbb{R}^m$ – вектор выхода, $p^2 \in \mathbb{R}^{s_2}$ – вектор параметров модели измерений, $h(t, x, p^2)$ – непрерывная вектор-функция. Возможные значения параметров модели измерения заданы брусом: $p^2 \in \boldsymbol{p}^2 \subset \mathbb{R}^{s_1}$. Множество $\boldsymbol{p}^2 = \boldsymbol{p}_1^2 \times \ldots \times \boldsymbol{p}_{s_2}^2$ характеризует неопределенность параметров модели измерений.

Управление, применяемое в каждый момент времени t, ищется в виде управления с неполной обратной связью: $u(t) = \boldsymbol{u}(t, z(t))$.

Множество допустимых управлений U образуют такие функции $\boldsymbol{u}(t, z)$, что $\forall t \in \boldsymbol{T} : u(t) = \boldsymbol{u}(t, z(t)) \in \boldsymbol{U}$, управление кусочно-непрерывно, а функция $f(t, x, \boldsymbol{u}(t, z), p^1)$ такова, что решение уравнения (2.19) существует и единственно.

Множество допустимых процессов $D(t_0, x_0)$ определяется как множество троек $d = (t_1, x(\cdot), u(\cdot))$, включающих момент t_1 окончания функционирования системы, траекторию $x(\cdot)$ и кусочно-непрерывное допустимое управление $u(\cdot)$, удовлетворяющих уравнению состояния (2.19), начальному условию (2.20) и конечному условию (2.21).

На множестве допустимых процессов $D(t_0, x_0)$ определен функционал качества управления

$$I(x_0, p^1, p^2, d) = \int_{t_0}^{t_1} f^0(t, x(t), u(t))dt + F(t_1, x(t_1)), \tag{2.23}$$

где $f^0(t, x, u)$, $F(t_1, x)$ – заданные непрерывные функции.

Для учёта конечных условий (2.21) к терминальному члену функционала (2.23) могут быть добавлены штрафные функции, характеризующие степень их невыполнения (см. разд. 2.1.1):

$$\bar{I}(x_0, p^1, p^2, d) = I(x_0, p^1, p^2, d) + \sum_{i=1}^{l} R_i^{(1)} \cdot H_i^{(1)} + \sum_{j=1}^{N_p} R_j^{(2)} \cdot H_j^{(2)}. \qquad (2.24)$$

Каждому допустимому управлению $\boldsymbol{u}(t,z) \in U$ и множествам $\boldsymbol{\Omega}$, $\boldsymbol{p}^1$ и $\boldsymbol{p}^2$ поставим в соответствие пучок (ансамбль) траекторий $x(t, \boldsymbol{u}(t, z(t)), x_0, p^1, p^2)$:

$$X(t, \boldsymbol{u}(t,z)) = \bigcup \{x(t, \boldsymbol{u}(t, z(t)), x_0, p^1, p^2) \mid x_0 \in \boldsymbol{\Omega}, p^1 \in \boldsymbol{p}^1, p^2 \in \boldsymbol{p}^2\}, \qquad (2.25)$$

т.е. объединение решений уравнения (2.19) по всем возможным начальным состояниям и параметрам модели объекта управления и модели измерений. Пучок траекторий порождается множествами $\boldsymbol{\Omega}$, $\boldsymbol{p}^1$, $\boldsymbol{p}^2$ и управлением $\boldsymbol{u}(t,z) \in U$.

Качество управления пучком траекторий предлагается оценивать величиной функционала

$$J[\boldsymbol{u}(t,z)] = \frac{\int_{\boldsymbol{p}^2} \int_{\boldsymbol{p}^1} \int_{\boldsymbol{\Omega}} \bar{I}(x_0, p^1, p^2, d) dx_0 dp^1 dp^2}{\text{mes}\, \boldsymbol{\Omega} \cdot \text{mes}\, \boldsymbol{p}^1 \cdot \text{mes}\, \mathbf{p}^2}, \qquad (2.26)$$

где $\text{mes}\, \boldsymbol{\Omega}, \text{mes}\, \boldsymbol{p}^1, \text{mes}\, \boldsymbol{p}^2$ – меры множеств $\boldsymbol{\Omega}, \boldsymbol{p}^1, \boldsymbol{p}^2$ соответственно.

Требуется найти такое управление $\boldsymbol{u}^*(t, x^1) \in U$, что

$$\boldsymbol{u}^*(t, x^1) = \text{Arg} \min_{\boldsymbol{u}(t,z) \in U} J[\boldsymbol{u}(t,z)]. \qquad (2.27)$$

Искомое управление называется оптимальным в среднем. Для решения поставленной задачи (2.27) предлагается задать структуру управления в параметрическом виде и, таким образом, свести задачу у конечномерной. Получаемое в результате управление является субоптимальным.

2.3.2. СТРАТЕГИЯ ПОИСКА УПРАВЛЕНИЯ

Будем предполагать, что:

- множество начальных состояний $\boldsymbol{\Omega}$ представляет собой брус $\boldsymbol{\Omega} = \boldsymbol{\omega}_1 \times \ldots \times \boldsymbol{\omega}_n$; каждый интервал $\boldsymbol{\omega}_i, i = 1, \ldots, n$ с помощью шага $\Delta_{\omega_i} \geq 0$ (при $\Delta_{\omega_i} = 0$ соответствующая компонента не разбивается, поскольку левая и правая границы интервалов совпадают, а интервал задает точку) разбивается на N_i непересекающихся интервалов, а брус $\boldsymbol{\Omega}$ делится на $N = N_1 \cdot \ldots \cdot N_n$ элементарных непересекающихся подмножеств $\boldsymbol{\Omega}_j$, $j = \overline{1, N}$; в каждом подмножестве $\boldsymbol{\Omega}_j$ задается начальное состояние x_0^j (центр $\boldsymbol{\Omega}_j$);

- аналогично брусы $\boldsymbol{p}^1$ и $\boldsymbol{p}^2$ разбиваются с помощью шагов $\Delta_{p^1} \geq 0$ и $\Delta_{p^2} \geq 0$ на $M^1 = M_1^1 \cdot \ldots \cdot M_{s_1}^1$ и $M^2 = M_1^2 \cdot \ldots \cdot M_{s_2}^2$ непересекающихся подмножеств $\boldsymbol{p}_{j_1}^1$ и $\boldsymbol{p}_{j_2}^2$ с центрами $p_{j_1}^1$ и $p_{j_2}^2$ соответственно,
- компоненты управления $\boldsymbol{u}(t,z) = (\boldsymbol{u}_1(t,z), \ldots, \boldsymbol{u}_q(t,z))^T$ ищутся в виде

$$\boldsymbol{u}_i(t,z) = \text{sat}\{g_i(t,z), \boldsymbol{U}_i\}, i = \overline{1,q}, \tag{2.28}$$

где $\text{sat}(g_i(t,z(t)), \boldsymbol{U}_i) = \begin{cases} \underline{\boldsymbol{U}_i}, g_i(t,z(t)) < \underline{\boldsymbol{U}_i}, \\ \overline{\boldsymbol{U}_i}, g_i(t,z(t)) > \overline{\boldsymbol{U}_i}, \\ g_i(t,z(t)), g_i(t,z(t)) \in \boldsymbol{U}_i, \end{cases}$ – функция насыщения;

- функции $g_i(t,z)$ предлагается искать в виде:

$$g_i(t,z) = \sum_{k_0=0}^{A_0} \sum_{k_1=0}^{A_1} \ldots \sum_{k_m=0}^{A_m} a_{k_0,k_1,\ldots k_m}^i \cdot \varphi_{k_0}^0(\tilde{t}) \cdot \varphi_{k_1}^1(\tilde{z}_1) \cdot \ldots \varphi_{k_m}^m(\tilde{z}_m), \tag{2.29}$$

где $a_{k_0,k_1,\ldots,k_m}^i, i = \overline{1,q}, k_0 = \overline{0,A_0}, \ldots, k_m = \overline{0,A_m}$ – неизвестные коэффициенты, $A_j, j = \overline{0,m}$ – масштабы усечения, $\{\varphi_{k_j}^j(\cdot)\}_{k_j=0}^{A_j}, j = \overline{0,m}$ – системы базисных функций по переменным, которые используются в управлении, $\tilde{t} = proj_{\boldsymbol{d}_0}^{\boldsymbol{e}_0}(t), \tilde{z}_1 = proj_{\boldsymbol{d}_1}^{\boldsymbol{e}_1}(z_1), \ldots, \tilde{z}_m = proj_{\boldsymbol{d}_m}^{\boldsymbol{e}_m}(z_m)$ – проекции значений времени и координат вектора состояния, используемых в управлении, на области определения соответствующих базисных функций, $\boldsymbol{e}_j, j = \overline{0,m}$ – оценки интервальной оболочки возможных значений переменной ($\omega(\boldsymbol{e}_j) \neq \infty, j = \overline{0,m}$), $\boldsymbol{d}_j, j = \overline{0,m}$ – области определения используемой системы базисных функций, $proj_{\boldsymbol{d}}^{\boldsymbol{e}}(x) = \begin{cases} (\omega(\boldsymbol{d}) / \omega(\boldsymbol{e})) \cdot (x - \underline{\boldsymbol{e}}) + \underline{\boldsymbol{d}}, & \omega(\boldsymbol{d}) \neq \infty, \\ (x - \underline{\boldsymbol{e}}) + \underline{\boldsymbol{d}}, & \underline{\boldsymbol{d}} \neq -\infty, \\ x, & \boldsymbol{d} =]-\infty; +\infty[\end{cases}$ – функции проекции величины x на область определения системы базисных функций.

В качестве систем базисных функций предлагаются полные системы ортонормированных функций с конечной областью определения $\boldsymbol{d} = [-1;1]$: полиномы Лежандра $P_n^L(x)$, Чебышева $P_n^C(x)$ и Гегенбауэра $P_{n,\alpha}^H(x)$, задаваемые формулами

$$P_n^L(x) = (2^n\ n!)^{-1} d^n (x^2-1)^n / dx^n,$$
$$P_n^C(x) = \sum_{k=0}^{\langle n/2 \rangle} C_n^{2\cdot k} (x^2-1)^k\ x^{n-2\cdot k}, \tag{2.30}$$
$$P_{n,\alpha}^H(x) = \sum_{j=0}^{\langle n/2 \rangle} (-1)^j\ (\alpha)_{n-j} / (j!\ (n-2\,j)!)\ (2\,x)^{n-2\,j},$$

где $\langle\cdot\rangle$ - целая часть числа, C_n^k - число сочетаний, α - параметр весовой функции, необходимый для определения скалярного произведения над полиномами Гегенбауэра, $(\alpha)_n = \Gamma(\alpha+n)/\Gamma(\alpha)$ - символ Похгаммера, $n = 0,1,...$

Стратегия поиска управления заключается в переходе от задачи (2.26) к конечномерной задаче поиска минимума приближенного значения функционала (2.26) с помощью подбора коэффициентов, образующих полиномы в (2.29). Для формализации задачи предлагается использовать вектор, являющийся гиперстолбцовой матрицей [66]:

$$a = \begin{pmatrix} \begin{pmatrix} \begin{pmatrix} a^1_{0,0,\ldots,0} \\ a^1_{1,0,\ldots,0} \\ \vdots \\ a^1_{A_0,0,\ldots,0} \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} a^1_{0,1,\ldots,0} \\ a^1_{1,1,\ldots,0} \\ \vdots \\ a^1_{A_0,1,\ldots,0} \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} a^1_{0,2,\ldots,0} \\ a^1_{1,2,\ldots,0} \\ \vdots \\ a^1_{A_0,2,\ldots,0} \end{pmatrix}^{\mathrm{T}} \cdots \begin{pmatrix} a^1_{0,A_1,\ldots,A_m} \\ a^1_{1,A_1,\ldots,A_m} \\ \vdots \\ a^1_{A_0,A_1,\ldots,A_m} \end{pmatrix}^{\mathrm{T}} \end{pmatrix}^{\mathrm{T}} \\ \vdots \\ \begin{pmatrix} \begin{pmatrix} a^q_{0,0,\ldots,0} \\ a^q_{1,0,\ldots,0} \\ \vdots \\ a^q_{A_0,0,\ldots,0} \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} a^q_{0,1,\ldots,0} \\ a^q_{1,1,\ldots,0} \\ \vdots \\ a^q_{A_0,1,\ldots,0} \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} a^q_{0,2,\ldots,0} \\ a^q_{1,2,\ldots,0} \\ \vdots \\ a^q_{A_0,2,\ldots,0} \end{pmatrix}^{\mathrm{T}} \cdots \begin{pmatrix} a^q_{0,A_1,\ldots,A_m} \\ a^q_{1,A_1,\ldots,A_m} \\ \vdots \\ a^q_{A_0,A_1,\ldots,A_m} \end{pmatrix}^{\mathrm{T}} \end{pmatrix}^{\mathrm{T}} \end{pmatrix} \tag{2.31}$$

Соответствующая интервальная гиперстолбцовая матрица:

$$\boldsymbol{a} = \begin{pmatrix} \begin{pmatrix} \begin{pmatrix} \boldsymbol{a}^1_{0,0,\ldots,0} \\ \boldsymbol{a}^1_{1,0,\ldots,0} \\ \vdots \\ \boldsymbol{a}^1_{A_0,0,\ldots,0} \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} \boldsymbol{a}^1_{0,1,\ldots,0} \\ \boldsymbol{a}^1_{1,1,\ldots,0} \\ \vdots \\ \boldsymbol{a}^1_{A_0,1,\ldots,0} \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} \boldsymbol{a}^1_{0,2,\ldots,0} \\ \boldsymbol{a}^1_{1,2,\ldots,0} \\ \vdots \\ \boldsymbol{a}^1_{A_0,2,\ldots,0} \end{pmatrix}^{\mathrm{T}} \cdots \begin{pmatrix} \boldsymbol{a}^1_{0,A_1,\ldots,A_m} \\ \boldsymbol{a}^1_{1,A_1,\ldots,A_m} \\ \vdots \\ \boldsymbol{a}^1_{A_0,A_1,\ldots,A_m} \end{pmatrix}^{\mathrm{T}} \end{pmatrix}^{\mathrm{T}} \\ \vdots \\ \begin{pmatrix} \begin{pmatrix} \boldsymbol{a}^q_{0,0,\ldots,0} \\ \boldsymbol{a}^q_{1,0,\ldots,0} \\ \vdots \\ \boldsymbol{a}^q_{A_0,0,\ldots,0} \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} \boldsymbol{a}^q_{0,1,\ldots,0} \\ \boldsymbol{a}^q_{1,1,\ldots,0} \\ \vdots \\ \boldsymbol{a}^q_{A_0,1,\ldots,0} \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} \boldsymbol{a}^q_{0,2,\ldots,0} \\ \boldsymbol{a}^q_{1,2,\ldots,0} \\ \vdots \\ \boldsymbol{a}^q_{A_0,2,\ldots,0} \end{pmatrix}^{\mathrm{T}} \cdots \begin{pmatrix} \boldsymbol{a}^q_{0,A_1,\ldots,A_m} \\ \boldsymbol{a}^q_{1,A_1,\ldots,A_m} \\ \vdots \\ \boldsymbol{a}^q_{A_0,A_1,\ldots,A_m} \end{pmatrix}^{\mathrm{T}} \end{pmatrix}^{\mathrm{T}} \end{pmatrix} \tag{2.32}$$

Таким образом, поставленная в разделе 2.3.1 задача сводится к проблеме нахождения наилучшего интервального гиперстолбцового вектора $\boldsymbol{a}$, минимизирующего функционал (2.26).

2.3.3. АЛГОРИТМ ПОИСКА УПРАВЛЕНИЯ

Для решения задачи поиска наилучшего вектора (2.31) и, как следствие, управления $\boldsymbol{u}(t,z)=(\boldsymbol{u}_1(t,z),\ldots,\boldsymbol{u}_q(t,z))^T$ далее применяются интервальные методы оптимизации, разработанные в главе 1. Вследствие того что данные алгоритмы применяются для оптимизации функций, требуется определить правило, ставящее в соответствие интервальному вектору коэффициентов $\boldsymbol{a}$ значение функционала (2.26). Зададим правило следующим образом:

1) на каждом элементарном множестве $\boldsymbol{\Omega}_{j_\Omega}$, $\boldsymbol{p}^1_{j_1}$, $\boldsymbol{p}^2_{j_2}$ задать $x_0^{j_\Omega}, p^1_{j_1}, p^2_{j_2}$ – начальное состояние и векторы параметров соответственно (как центральные точки $\boldsymbol{\Omega}_{j_\Omega}$, $\boldsymbol{p}^1_{j_1}$, $\boldsymbol{p}^2_{j_2}$) для каждого j_Ω, j_1, j_2;
2) по вектору $\boldsymbol{a}$ вида (2.32), размер которого определяется заданными масштабами усечения A_j, $j=\overline{0,m}$, найти соответствующее интервальное управление $\boldsymbol{u}(t,z)$;
3) найти решение $\boldsymbol{x}^{j_\Omega,j_1,j_2}(t)$ уравнения модели (2.19) с управлением $\boldsymbol{u}(t,z)$ и начальным состоянием $x_0^{j_\Omega}$ и векторами параметров $p^1_{j_1}, p^2_{j_2}$ для каждого j_Ω, j_1, j_2 на промежутке времени $[t_o; t_1^{j_\Omega,j_1,j_2}]$, правый конец которого удовлетворяет (2.21);
4) вычислить интервальное значение функционала (2.24): $\bar{I}(x_0^{j_\Omega}, p^1_{j_1}, p^2_{j_2}, d^{j_\Omega,j_1,j_2})$, где $d^{j_\Omega,j_1,j_2}=(t_1^{j_\Omega,j_1,j_2}, \boldsymbol{x}^{j_\Omega,j_1,j_2}(t), \boldsymbol{u}(t)=\boldsymbol{u}(t,h(t,\boldsymbol{x}^{j_\Omega,j_1,j_2}(t),p^2_{j_2})))$ для каждого j_Ω, j_1, j_2;
5) после нахождения функционала (2.24) для всех начальных состояний и всех значений компонент векторов параметров моделей объекта управления и измерений вычислить интервальное значение критерия качества управления пучком траекторий:

$$J[\boldsymbol{u}(t,z)]=\frac{\sum\limits_{j_q=1}^{L}\sum\limits_{j_p=1}^{M}\sum\limits_{j_\Omega=1}^{N}\bar{I}(x_0^{j_\Omega}, p^{j_p}, q^{j_q}, d^{j_\Omega,j_p,j_q})}{L\cdot M\cdot N}. \qquad (2.33)$$

Величина $J[\boldsymbol{u}(t,z)]$ ставится в соответствие вектору $\boldsymbol{a}$ и обозначается как $\tilde{J}(\boldsymbol{a})$. В качестве приближенного решения задачи выбирается наилучший вектор $\boldsymbol{a}^*$ и соответствующее ему интервальное управление $\boldsymbol{u}^*(t,z)$. Для нахождения вектора a^* предлагается использовать интервальные алгоритмы оптимизации, описанные в главе 1.

2.4. ЗАКЛЮЧЕНИЕ

Во второй главе:

1) приведены постановки трех задач синтеза оптимального управления,
2) разработаны интервальные алгоритмы поиска оптимального программного управления нелинейными непрерывными детерминированными системами,
3) разработаны интервальные алгоритмы синтеза оптимального управления с неполной обратной связью нелинейными непрерывными детерминированными системами,
4) разработаны интервальные алгоритмы синтеза оптимального управления по выходу нелинейными непрерывными детерминированными системами.

ГЛАВА 3. ПРОГРАММНЫЙ КОМПЛЕКС «ИНТЕРВАЛЬНЫЕ МЕТОДЫ ОПТИМИЗАЦИИ НЕЛИНЕЙНЫХ ДЕТЕРМИНИРОВАННЫХ СИСТЕМ»

Описанные в главах 1, 2 алгоритмы реализованы в виде программного комплекса «Интервальные методы оптимизации нелинейных детерминированных систем». Комплекс разработан в среде Microsoft Visual Studio 2015, на языке C#. Он содержит два основных модуля алгоритмов оптимизации и модуль поддержки, в котором реализованы следующие элементы: интервальные арифметик, алгоритм инвертера, интервальные расширения и процедуры интегрирования. В первом реализованы два класса интервальных алгоритмов глобальной условной оптимизации (на основе инвертера и метаэвристические). Во втором – четыре блока основных решаемых задач. Общая схема комплекса представлена на рис. 3.1.

Рис. 3.1. Структура программного комплекса

В блоке «Инверсные методы» реализованы алгоритмы из раздела 1.2. Примеры окон программного комплекса, реализующего данные алгоритмы, представлены на рисунках 3.2 – 3.4.

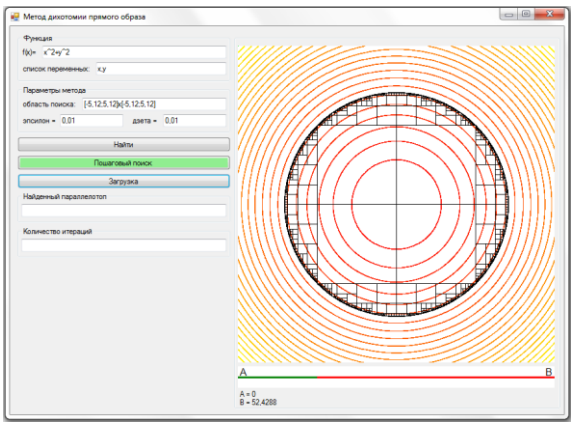


Рис. 3.2. Метод дихотомии целевого интервала

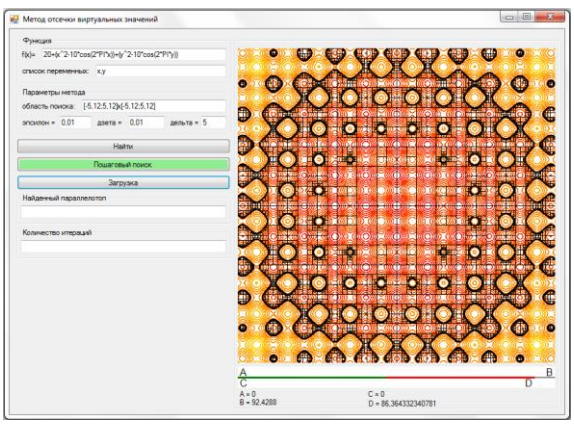


Рис. 3.3. Метод отсечки виртуальных значений

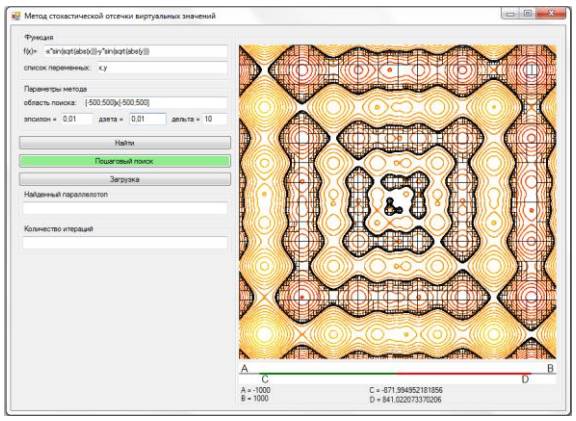


Рис. 3.4. Метод стохастической отсечки виртуальных значений

В блоке «Метаэвристические методы» реализованы алгоритмы из раздела 1.3. Примеры окон программного комплекса, реализующего данные алгоритмы, представлены на рисунках 3.5, 3.6.

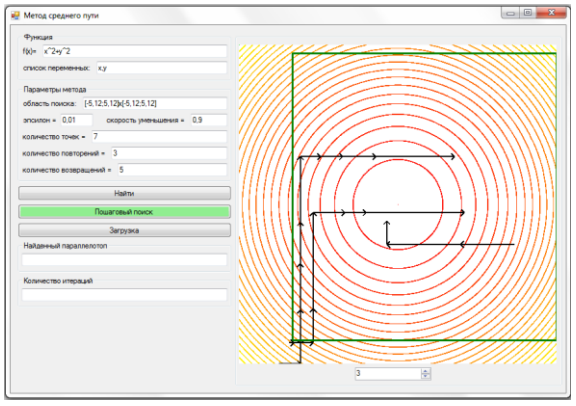


Рис. 3.5. Метод среднего пути

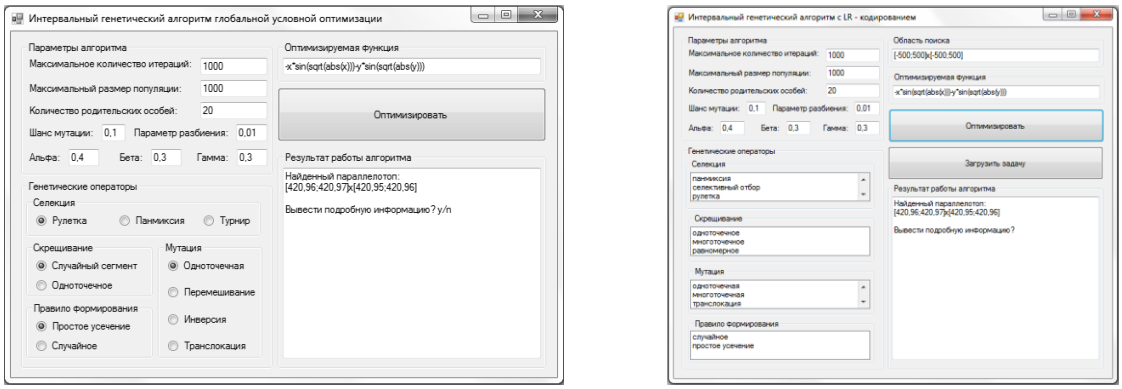


Рис. 3.6. Интервальный генетический алгоритм

На рис. 3.2 – 3.6 представлены графические окна разработанных интервальных алгоритмов глобальной условной оптимизации, которые содержат окна для ввода параметров, а также текстовые или графические окна для представления промежуточных или финальных результатов работы алгоритмов.

В последнем блоке реализованы алгоритмы из главы 2. Примеры окон программного комплекса, реализующего данные алгоритмы, представлены ниже.

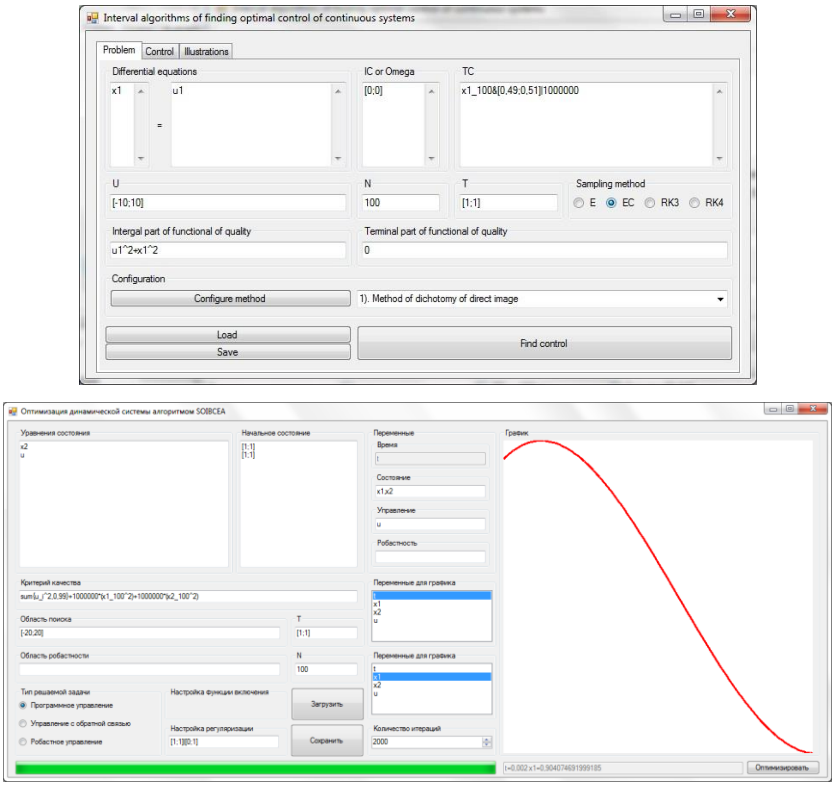


Рис. 3.7. Поиск оптимального программного управление

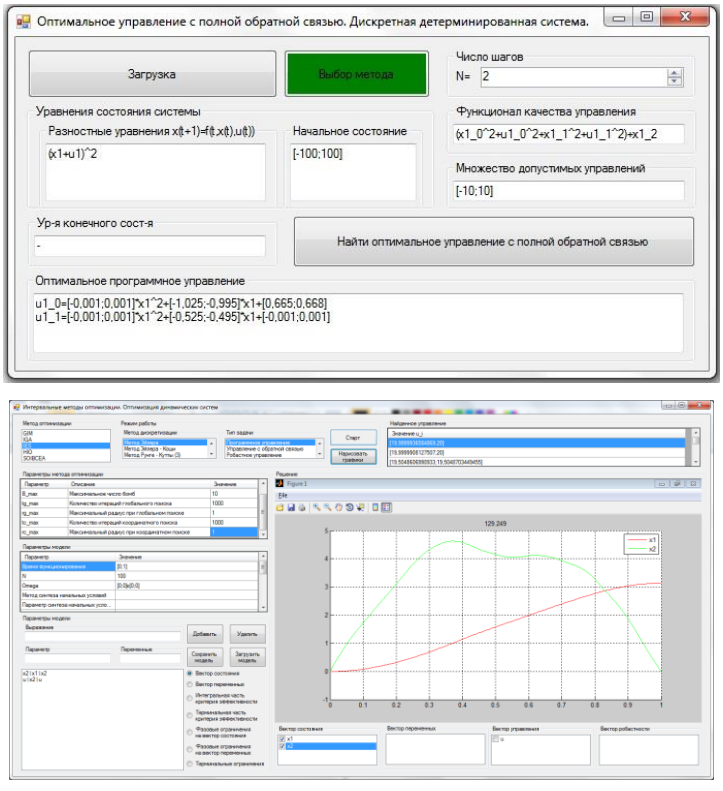


Рис. 3.8. Синтез оптимального управления с обратной связью

Разработанный программный комплекс был протестирован при решении модельных и прикладных задач оптимизации технических систем и управления авиационно-космическими системами. Подробные результаты изложены в следующей главе.

ГЛАВА 4. ПРИЛОЖЕНИЕ ИНТЕРВАЛЬНЫХ МЕТОДОВ В ЗАДАЧАХ ОПТИМИЗАЦИИ ТЕХНИЧЕСКИХ СИСТЕМ И УПРАВЛЕНИЯ АВИАЦИОННО-КОСМИЧЕСКИМИ СИСТЕМАМИ

В четвертой главе решены модельные и прикладные задачи:

- оптимизации технических систем:
  - определение параметров сварной балки (целью является определение минимальной по стоимости конструкции балки, удовлетворяющей ограничениям по напряжению сдвига, изгиба, продольной нагрузке и отклонению края),
  - определение параметров сосуда высокого давления (целью является определение параметров баллона для хранения сжатого газа, минимизировав его стоимость),
  - определение параметров редуктора (целью является определение минимальной по весу конструкции редуктора, при этом конструкция редуктора должна удовлетворять ограничениям по напряжению изгиба зубцов шестерни, поверхностному напряжению, поперечным отклонениям валов и напряжению на валах),
  - определение параметров натяжной/компрессионной пружины (целью является определение минимальной по весу конструкции пружины, ограниченной по минимальному отклонению, напряжению сдвига, частоте колебаний и ограничениям на внешний диаметр),
- оптимального управления объектами авиационно-космической техники:
  - задача преследования (трехмерная задача реализации маневра наведения),
  - управление солнечным парусом (поиск оптимального по быстродействию управления для межпланетной миссии Земля-Меркурий),
  - стабилизация спутника (задача гашения вращательного движения спутника с помощью установленных на нем двигателей),
  - задача перехвата (задача поиска терминального управления, в ходе которой реализуется маневр наведения),
  - задача групповой навигации (вариант задачи перехвата с несколькими перехватчиками и целями),

- приземление гиперзвукового летательного аппарата (управление ГЗЛА на скоростях более 5М).

4.1. ЗАДАЧИ ОПТИМИЗАЦИИ ТЕХНИЧЕСКИХ СИСТЕМ

Во всех задачах данного раздела были использованы следующие параметры интервальных методов глобальной оптимизации:

- параметры адаптивного интервального алгоритма: $N_{\max} = 2500, \gamma = 0,98, w_{init} = 0,8$, $N_B = 8, \gamma_B = 0,98, C_B^+ = C_B^- = 4, \alpha_B = \beta_B = 0,4$, $r^B = 1$, $q^+ = q^- = (0,4;0,3;0,2;0,1)^{\mathrm{T}}$, $N_W = 5, Sp = (10;10)^{\mathrm{T}}$, $\rho_W = 0,2, \gamma_W = 0,9$, $A_{good} = 3$, $A_{bad} = 2$, $N_A = 10, \gamma_A = 0,98, A_{\max} = 10, r^A = 1$;
- параметры интервального генетического алгоритма: $t_{\max} = 1000, \varepsilon = 0,001, pop_{amount} = 100, pool_{amount} = 20, r = 0,025$, LR кодирование, одноточечные скрещивание и мутация, рулетка, простое усечение;
- параметры интервального метода взрывов: $t_{\max}^g = t_{\max}^c = 2500, B_{\max} = 100, r_{\max} = 1$.

4.1.1. ЗАДАЧА ОПРЕДЕЛЕНИЯ ПАРАМЕТРОВ СВАРНОЙ БАЛКИ

В рассматриваемой задаче требуется определить параметры конструкции сварной балки исходя из имеющихся ограничений [113].

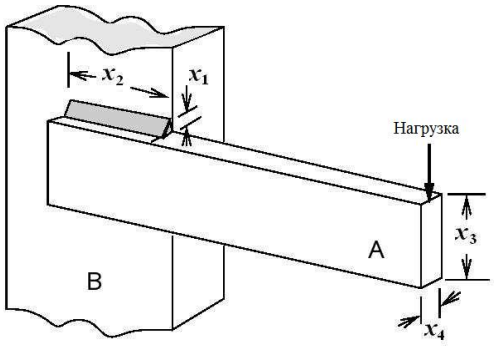


Рис. 4.1. Сварная балка

Рассматриваемая конструкция состоит из балки A и области сварки, необходимой для ее прикрепления к балке B. Целью является определение минимальной по стоимости конструкции балки, описываемой вектором параметров $x = (x_1, x_2, x_3, x_4)^T$. Кроме того, балка должна удовлетворять ограничениям по напряжению сдвига τ, изгиба σ, продольной нагрузке P_c и отклонению края балки δ. Задача может быть формализована следующим образом:

$$f(t) = 1{,}10471 \cdot x_1^2 \cdot x_2 + 0{,}04811 \cdot x_3 \cdot x_4 \cdot (14 + x_2),$$
$$g^1(x) = \tau(x) - 13600 \le 0,$$
$$g^2(x) = \sigma(x) - 30000 \le 0,$$
$$g^3(x) = x_1 - x_4 \le 0,$$
$$g^4(x) = 0{,}10471 \cdot x_1^2 + 0{,}04811 \cdot x_3 \cdot x_4 \cdot (14 + x_2) - 5 \le 0,$$
$$g^5(x) = 0{,}125 - x_1 \le 0,$$
$$g^6(x) = \delta(x) - 0{,}25 \le 0,$$
$$g^7(x) = 6000 - P_c(x) \le 0,$$
$$\boldsymbol{s} = [0{,}1; 2{,}0] \times [0{,}1; 10{,}0] \times [0{,}1; 10{,}0] \times [0{,}1; 2{,}0],$$

где $\tau(x) = \sqrt{(\tau')^2 + (2 \cdot \tau' \cdot \tau'') \cdot \frac{x_2}{2 \cdot R} + (\tau'')^2}$, $\tau' = \frac{6000}{\sqrt{2} \cdot x_1 \cdot x_2}$, $\tau'' = \frac{M \cdot R}{J}$, $M = 6000 \cdot (14 + \frac{x_2}{2})$,

$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}$, $J = 2 \cdot \sqrt{2} \cdot x_1 \cdot x_2 \cdot (\frac{x_2^2}{12} + (\frac{x_1 + x_3}{2})^2)$, $\sigma(x) = \frac{50400}{x_3^2 \cdot x_4}$, $\delta(x) = \frac{65{,}856}{30 \cdot x_3^3 \cdot x_4}$,

$P_c(x) = \frac{4{,}013 \cdot 5 \cdot 10^6 \cdot \sqrt{x_3^2 \cdot x_4^6}}{196} \cdot (1 - \frac{x_3 . \sqrt{5/8}}{28})$.

Вспомогательная функция выглядит следующим образом:

$$\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}) + \sum_{j=1}^{7} 10^7 \cdot h_\infty^0(\boldsymbol{g}^j(\boldsymbol{x}),]-\infty; 0]).$$

Решение, найденное в [84], и соответствующее ему значение целевой функции и ограничений: $x^* = (0{,}20384; 3{,}52938; 9{,}03629; 0{,}20586)^T$, $f(x^*) = 1{,}724852$, $g^1(x^*) = -0{,}25400$, $g^2(x^*) = 0{,}092700$, $g^3(x^*) = 0{,}000001$, $g^4(x^*) = -3{,}432989$, $g^5(x^*) = -0{,}080730$, $g^6(x^*) = -0{,}235540$, $g^7(x^*) = 0{,}055938$.

Табл. 4.1. Результаты работы алгоритмов

Величина	IES	IGA	AIA
$\tilde{\boldsymbol{x}}$	$\begin{pmatrix}[0{,}20386;0{,}20387]\\ [3{,}52933;3{,}52939]\\ [9{,}03598;9{,}03605]\\ [0{,}20581;0{,}20586]\end{pmatrix}$	$\begin{pmatrix}[0{,}20381;0{,}20383]\\ [3{,}52940;3{,}52941]\\ [9{,}03625;9{,}03627]\\ [0{,}20586;0{,}20587]\end{pmatrix}$	$\begin{pmatrix}[0{,}20390;0{,}20392]\\ [3{,}52930;3{,}52935]\\ [9{,}03601;9{,}03602]\\ [0{,}20589;0{,}20595]\end{pmatrix}$
$\boldsymbol{f}(\tilde{\boldsymbol{x}})$	$[1{,}7303; 1{,}7308]$	$[1{,}7307; 1{,}7309]$	$[1{,}7311; 1{,}7315]$
$\boldsymbol{g}^1(\tilde{\boldsymbol{x}})$	$[-55{,}9922; -54{,}7761]$	$[-53{,}5217; -52{,}0211]$	$[-59{,}1598; -57{,}4810]$
$\boldsymbol{g}^2(\tilde{\boldsymbol{x}})$	$[-15{,}1889; -7{,}4396]$	$[-18{,}1053; -16{,}5161]$	$[-28{,}0932; -19{,}2925]$
$\boldsymbol{g}^3(\tilde{\boldsymbol{x}})$	$[-0{,}0020; -0{,}0019]$	$[-0{,}0021; -0{,}0020]$	$[-0{,}0021; -0{,}0020]$

$\boldsymbol{g}^4(\tilde{\boldsymbol{x}})$	$[-3{,}4273; -3{,}4269]$	$[-3{,}4268; -3{,}4267]$	$[-3{,}4267; -3{,}4262]$
$\boldsymbol{g}^5(\tilde{\boldsymbol{x}})$	$[-0{,}0789; -0{,}0789]$	$[-0{,}0788; -0{,}0787]$	$[-0{,}0789; -0{,}0789]$
$\boldsymbol{g}^6(\tilde{\boldsymbol{x}})$	$[-0{,}2355; -0{,}2355]$	$[-0{,}2356; -0{,}2355]$	$[-0{,}2356; -0{,}2356]$
$\boldsymbol{g}^7(\tilde{\boldsymbol{x}})$	$[-11{,}1778; -6{,}7364]$	$[-12{,}1387; -11{,}2448]$	$[-19{,}0385; -13{,}7705]$

4.1.2. ЗАДАЧА ОПРЕДЕЛЕНИЯ ПАРАМЕТРОВ СОСУДА ВЫСОКОГО ДАВЛЕНИЯ

В рассматриваемой задаче требуется определить параметры баллона для хранения сжатого газа [116].

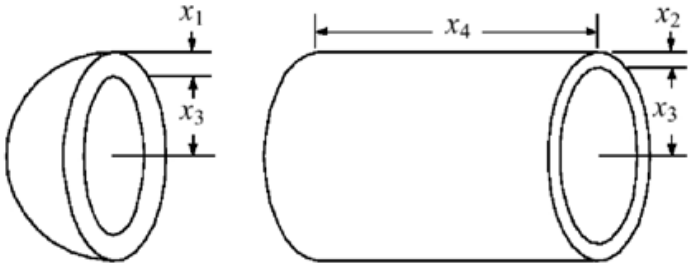


Рис. 4.2. Сосуд высокого давления

Целью является определение минимальной по стоимости конструкции сосуда, описываемой вектором параметров $x = (x_1, x_2, x_3, x_4)^T$, соответствующих толщине, толщине головки, внутреннему радиусу и длине цилиндрической части. Кроме того, величины x_1 и x_2 являются дискретными величинами (описывающими кратность параметра величине $0{,}0625$). Задача может быть формализована следующим образом:

$$f(x) = 0{,}6224 \cdot \tilde{x}_1 \cdot x_3 \cdot x_4 + 1{,}7781 \cdot \tilde{x}_2 \cdot x_3^2 + 3{,}1661 \cdot \tilde{x}_1^2 \cdot x_4 + 19{,}84 \cdot \tilde{x}_1^2 \cdot x_3,$$

$$g^1(x) = -\tilde{x}_1 + 0{,}0193 \cdot x_3 \le 0,$$

$$g^2(x) = -\tilde{x}_2 + 0{,}00954 \cdot x_3 \le 0,$$

$$g^3(x) = -\pi \cdot x_3^2 \cdot x_4 - \frac{4}{3} \cdot \pi \cdot x_3^3 + 1296000 \le 0,$$

$$g^4(x) = x_4 - 240 \le 0,$$

$$\boldsymbol{s} = [1; 99{,}99] \times [1; 99{,}99] \times [10; 200] \times [10; 200],$$

где $\tilde{x}_1 = 0{,}0625 \cdot \langle x_1 \rangle$, $\tilde{x}_2 = 0{,}0625 \cdot \langle x_2 \rangle$, $\langle \cdot \rangle$ – целая часть числа.

Вспомогательная функция выглядит следующим образом: $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}) + \sum_{j=1}^{7} 10^7 \cdot h_\infty^0(\boldsymbol{g}^j(\boldsymbol{x}),]-\infty; 0])$.

Решение, найденное в [84], и соответствующее ему значение целевой функции и ограничений: $x^* = (13; 7; 42,098446; 176,636596)^T$, $f(x^*) = 6059,714335$, $g^1(x^*) = 0,000000$, $g^2(x^*) = -0,035881$, $g^3(x^*) = -0,028761$, $g^4(x^*) = -63,363404$.

Табл. 4.2. Результаты работы алгоритмов

Величина	IES	IGA	AIA
$\tilde{\boldsymbol{x}}$	[13,0000;13,0002] [7,0000;7,0003] [42,1130;42,1137] [176,4812;176,4830]	[13,0000;13,0001] [7,0000;7,0002] [42,1139;42,1141] [176,4586;176,4587]	[13,0000;13,0003] [7,0000;7,0003] [42,1129;42,1138] [176,4851;176,4861]
$\boldsymbol{f}(\tilde{\boldsymbol{x}})$	[6058,5242; 6058,6839]	[6058,1468; 6058,1827]	[6058,5987; 6058,7732]
$\boldsymbol{g}^1(\tilde{\boldsymbol{x}})$	[0,0002; 0,0003]	[0,0002; 0,0003]	[0,0003; 0,0003]
$\boldsymbol{g}^2(\tilde{\boldsymbol{x}})$	[–0,0357; –0,0357]	[–0,0358; –0,0357]	[–0,0357; –0,0357]
$\boldsymbol{g}^3(\tilde{\boldsymbol{x}})$	[–196,9021; –138,5834]	[–89,0990; –74,7457]	[–221,0734; –153,4142]
$\boldsymbol{g}^4(\tilde{\boldsymbol{x}})$	[–63,5188; –63,5170]	[–63,5414; –63,5413]	[–63,5149; –63,5139]

4.1.3. ЗАДАЧА ОПРЕДЕЛЕНИЯ ПАРАМЕТРОВ РЕДУКТОРА

В рассматриваемой задаче требуется определить параметры редуктора [91].

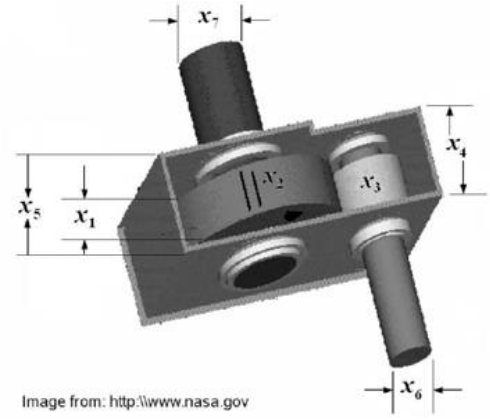


Рис. 4.3. Конструкция редуктора

Целью является определение минимальной по весу конструкции редуктора, описываемой вектором параметров $x = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)^T$, соответствующих ширине лицевой стороны, длине зубцов, числу зубцов на шестерне (целочисленная величина), длине первого вала, длине второго вала, диаметру первого вала и диаметру второго вала. Конструкция редуктора должна удовлетворять ограничениям по напряжению изгиба зубцов

шестерни, поверхностному напряжению, поперечным отклонениям валов и напряжению на валах. Задача может быть формализована следующим образом:

$$f(t) = 0,7854 \cdot x_1 \cdot x_2^2 \cdot (3,3333 \cdot \langle x_3 \rangle^2 + 14,9334 \cdot \langle x_3 \rangle - 43,0934) -$$
$$-1,508 \cdot x_1 \cdot (x_6^2 + x_7^2) + 7,4777 \cdot (x_6^3 + x_7^3) + 0,7854 \cdot (x_4 \cdot x_6^2 + x_5 \cdot x_7^2),$$

$$g^1(x) = \frac{27}{x_1 \cdot x_2^2 \cdot \langle x_3 \rangle} - 1 \le 0,$$

$$g^2(x) = \frac{397,5}{x_1 \cdot x_2^2 \cdot \langle x_3 \rangle^2} - 1 \le 0,$$

$$g^3(x) = \frac{1,93 \cdot x_4^3}{x_2 \cdot \langle x_3 \rangle \cdot x_6^4} - 1 \le 0,$$

$$g^4(x) = \frac{1,93 \cdot x_5^3}{x_2 \cdot \langle x_3 \rangle \cdot x_7^4} - 1 \le 0,$$

$$g^5(x) = \frac{1}{110 \cdot x_6^3} \cdot \sqrt{\left(\frac{745 \cdot x_4}{x_2 \cdot \langle x_3 \rangle} \right)^2 + 16,9 \cdot 10^6} - 1 \le 0,$$

$$g^6(x) = \frac{1}{85 \cdot x_7^3} \cdot \sqrt{\left(\frac{745 \cdot x_5}{x_2 \cdot \langle x_3 \rangle} \right)^2 + 157,5 \cdot 10^6} - 1 \le 0,$$

$$g^7(x) = \frac{x_2 \cdot \langle x_3 \rangle}{40} - 1 \le 0,$$

$$g^8(x) = \frac{5 \cdot x_2}{x_1} - 1 \le 0,$$

$$g^9(x) = \frac{x_1}{12 \cdot x_2} - 1 \le 0,$$

$$g^{10}(x) = \frac{1,5 \cdot x_6 + 1,9}{x_5} - 1 \le 0,$$

$$g^{11}(x) = \frac{1,1 \cdot x_7 + 1,9}{x_5} - 1 \le 0,$$

$$\boldsymbol{s} = [2,6; 3,6] \times [0,7; 0,8] \times [17; 28,99] \times [7,3; 8,3] \times [7,8; 8,3] \times [2,9; 3,9] \times [5,0; 5,5],$$

где $\langle \cdot \rangle$ - целая часть числа.

Вспомогательная функция выглядит следующим образом: $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}) + \sum_{j=1}^{11} 10^7 \cdot h_\infty^0(\boldsymbol{g}^j(\boldsymbol{x}),]-\infty; 0])$.

Решение, найденное в [84], и соответствующее ему значение целевой функции и ограничений: $x^* = (3,5; 0,7; 17; 7,3; 7,8; 3,350214; 5,286683)^T$, $f(x^*) = 2996,348165$, $g^1(x^*) = -0,073915$, $g^2(x^*) = -0,197999$, $g^3(x^*) = -0,499172$, $g^4(x^*) = -0,901472$,

$g^5(x^*) = 6 \cdot 10^{-7}$, $g^6(x^*) = 1{,}3 \cdot 10^{-7}$, $g^7(x^*) = -0{,}702500$, $g^8(x^*) = 0{,}0$, $g^9(x^*) = -0{,}583333$, $g^{10}(x^*) = -0{,}112138$, $g^{11}(x^*) = -0{,}010852$.

Табл. 4.3. Результаты работы алгоритмов

Величина	IES	IGA	AIA
$\tilde{\boldsymbol{x}}$	$\begin{pmatrix} [3{,}4999; 3{,}5002] \\ [0{,}7000; 0{,}7001] \\ [17{,}0000; 17{,}0003] \\ [7{,}3021; 7{,}3023] \\ [7{,}8001; 7{,}8002] \\ [3{,}3501; 3{,}3503] \\ [5{,}2864; 5{,}2865] \end{pmatrix}$	$\begin{pmatrix} [3{,}4995; 3{,}4999] \\ [0{,}7000; 0{,}7001] \\ [17{,}0000; 17{,}0001] \\ [7{,}3018; 7{,}3019] \\ [7{,}7999; 7{,}8002] \\ [3{,}3503; 3{,}3504] \\ [5{,}2861; 5{,}2865] \end{pmatrix}$	$\begin{pmatrix} [3{,}4999; 3{,}5002] \\ [0{,}7000; 0{,}7001] \\ [17{,}0000; 17{,}0003] \\ [7{,}3021; 7{,}3023] \\ [7{,}8001; 7{,}8002] \\ [3{,}3501; 3{,}3503] \\ [5{,}2864; 5{,}2865] \end{pmatrix}$
$\boldsymbol{f}(\tilde{\boldsymbol{x}})$	$[2995{,}1388; 2997{,}2658]$	$[2995{,}7668; 2996{,}7621]$	$[2996{,}0899; 2996{,}8389]$
$\boldsymbol{g}^1(\tilde{\boldsymbol{x}})$	$[-0{,}0748; -0{,}0739]$	$[-0{,}0742; -0{,}0738]$	$[-0{,}0742; -0{,}0738]$
$\boldsymbol{g}^2(\tilde{\boldsymbol{x}})$	$[-0{,}1988; -0{,}1979]$	$[-0{,}1982; -0{,}1979]$	$[-0{,}1982; -0{,}1979]$
$\boldsymbol{g}^3(\tilde{\boldsymbol{x}})$	$[-0{,}4989; -0{,}4985]$	$[-0{,}4990; -0{,}4988]$	$[-0{,}4988; -0{,}4986]$
$\boldsymbol{g}^4(\tilde{\boldsymbol{x}})$	$[-0{,}9014; -0{,}9013]$	$[-0{,}9015; -0{,}9014]$	$[-0{,}9015; -0{,}9014]$
$\boldsymbol{g}^5(\tilde{\boldsymbol{x}})$	$[0{,}0001; 0{,}0003]$	$[-0{,}0001; -0{,}0001]$	$[-0{,}0001; 0{,}0001]$
$\boldsymbol{g}^6(\tilde{\boldsymbol{x}})$	$[0{,}0006; 0{,}0009]$	$[0{,}0001; 0{,}0003]$	$[0{,}0001; 0{,}0002]$
$\boldsymbol{g}^7(\tilde{\boldsymbol{x}})$	$[-0{,}7025; -0{,}7024]$	$[-0{,}7025; -0{,}7024]$	$[-0{,}7025; -0{,}7024]$
$\boldsymbol{g}^8(\tilde{\boldsymbol{x}})$	$[-0{,}0001; 0{,}0004]$	$[0{,}0000; 0{,}0003]$	$[-0{,}0001; 0{,}0002]$
$\boldsymbol{g}^9(\tilde{\boldsymbol{x}})$	$[-0{,}5835; -0{,}5833]$	$[-0{,}5834; -0{,}5833]$	$[-0{,}5834; -0{,}5833]$
$\boldsymbol{g}^{10}(\tilde{\boldsymbol{x}})$	$[-0{,}1122; -0{,}1122]$	$[-0{,}1121; -0{,}1121]$	$[-0{,}1122; -0{,}1121]$
$\boldsymbol{g}^{11}(\tilde{\boldsymbol{x}})$	$[-0{,}0111; -0{,}0110]$	$[-0{,}0111; -0{,}0109]$	$[-0{,}0109; -0{,}0108]$

4.1.4. ЗАДАЧА ОПРЕДЕЛЕНИЯ ПАРАМЕТРОВ НАТЯЖНОЙ/КОМПРЕССИОННОЙ ПРУЖИНЫ

В рассматриваемой задаче требуется определить параметры натяжной/компрессионной пружины, учитывая физические ограничения [94].

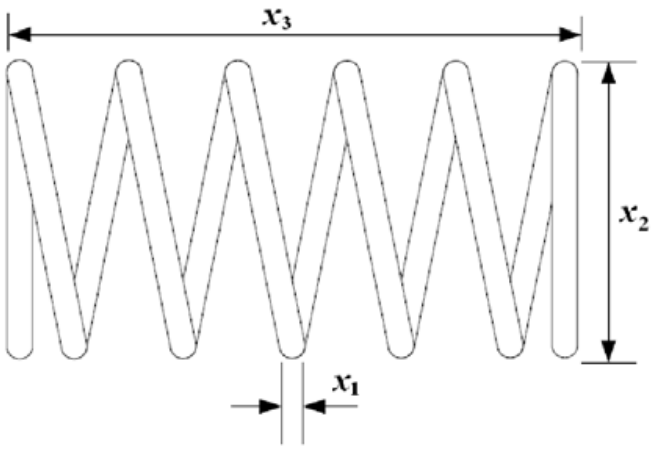


Рис. 4.4. Конструкция пружины

Целью является определение минимальной по весу конструкции пружины, описываемой вектором параметров $x = (x_1, x_2, x_3)^T$, соответствующих диаметру проволоки, среднему диаметру витка и числу активных витков. Конструкция пружины должна удовлетворять ограничениям по минимальному отклонению, напряжению сдвига, частоте колебаний и ограничениям на внешний диаметр. Задача может быть формализована следующим образом:

$$f(t) = (x_3 + 2) \cdot x_1^2 \cdot x_2,$$

$$g^1(x) = 1 - \frac{x_2^3 \cdot x_3}{71875 \cdot x_1^4} \le 0,$$

$$g^2(x) = \frac{4 \cdot x_2^2 - x_1 \cdot x_2}{12566 \cdot \left(x_1^3 \cdot x_2 - x_1^4\right)} + \frac{2,46}{12566 \cdot x_1^2} - 1 \le 0,$$

$$g^3(x) = 1 - \frac{140,54 \cdot x_1}{x_2^2 \cdot x_3} \le 0,$$

$$g^4(x) = \frac{x_1 + x_2}{1,5} - 1 \le 0,$$

$$\boldsymbol{s} = [0,05; 2,0] \times [0,25; 1,3] \times [2,0; 15,0].$$

Вспомогательная функция выглядит следующим образом: $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}) + \sum_{j=1}^{4} 10^7 \cdot h_\infty^0(\boldsymbol{g}^j(\boldsymbol{x}),]-\infty; 0])$.

Решение, найденное в [84], и соответствующее ему значение целевой функции и ограничений: $x^* = (0,05158; 0,35419; 11,4387)^T$, $f(x^*) = 0,012665$, $g^1(x^*) = -9,001053$, $g^2(x^*) = 0,000020$, $g^3(x^*) = -4,057026$, $g^4(x^*) = -0,727707$.

Табл. 4.4. Результаты работы алгоритмов

Величина	IES	IGA	AIA
$\tilde{\boldsymbol{x}}$	$\begin{pmatrix}[0,0514;0,0515]\\[0,3541;0,3542]\\[11,4384;11,4386]\end{pmatrix}$	$\begin{pmatrix}[0,0509;0,0513]\\[0,3537;0,3539]\\[11,4378;11,4379]\end{pmatrix}$	$\begin{pmatrix}[0,0511;0,0517]\\[0,3541;0,3545]\\[11,4385;11,4390]\end{pmatrix}$
$\boldsymbol{f}(\tilde{\boldsymbol{x}})$	$[0,0126;0,0126]$	$[0,0123;0,0125]$	$[0,0124;0,0127]$
$\boldsymbol{g}^1(\tilde{\boldsymbol{x}})$	$[-0,01445;-0,0057]$	$[-0,0522;-0,0180]$	$[-0,0412;-0,0097]$
$\boldsymbol{g}^2(\tilde{\boldsymbol{x}})$	[0,0028;0,01164]	[0,0008;0,0433]	[-0,0149;0,0366]
$\boldsymbol{g}^3(\tilde{\boldsymbol{x}})$	$[-4,0432;-4,0305]$	$[-4,0353;-3,9904]$	$[-4,0062;-3,9925]$
$\boldsymbol{g}^4(\tilde{\boldsymbol{x}})$	$[-0,7297;-0,7295]$	$[-0,7303;-0,7299]$	$[-0,7299;-0,7292]$

4.2. ЗАДАЧИ ОПТИМАЛЬНОГО УПРАВЛЕНИЯ АВИАЦИОННО-КОСМИЧЕСКИМИ СИСТЕМАМИ

4.2.1. ЗАДАЧА ПРЕСЛЕДОВАНИЯ

Рассмотрим решение задачи преследования в трехмерном пространстве [121]. Поведение модели объекта управления описывается системой дифференциальных уравнений (2.1):

$$\dot{x}(t)=V_x(t),\ \ \dot{V}_x(t)=u_x(t),\ \ \dot{x}^t(t)=V_x{}^t(t),\ \ \dot{V}_x{}^t(t)=u_x{}^t(t),$$
$$\dot{y}(t)=V_y(t),\ \ \dot{V}_y(t)=u_y(t),\ \ \dot{y}^t(t)=V_y{}^t(t),\ \ \dot{V}_y{}^t(t)=u_y{}^t(t),$$
$$\dot{z}(t)=V_z(t),\ \ \dot{V}_z(t)=u_z(t),\ \ \dot{z}^t(t)=V_z{}^t(t),\ \ \dot{V}_z{}^t(t)=u_z{}^t(t),$$

где $r=(x,y,z)^{\mathrm{T}}$ и $r^t=(x^t,y^t,z^t)^{\mathrm{T}}$ - радиус-векторы, описывающие положение перехватчика и цели соответственно; $V=(V_x,V_y,V_z)^{\mathrm{T}}$ и $V^t=(V_x{}^t,V_y{}^t,V_z{}^t)^{\mathrm{T}}$ - векторы скоростей перехватчика и цели соответственно; $u=(u_x,u_y,u_z)^{\mathrm{T}}$ и $u^t=(u_x{}^t,u_y{}^t,u_z{}^t)^{\mathrm{T}}$ - ускорения перехватчика (управления) и цели соответственно. Предполагается, что цель движется с постоянным ускорением $u^t=(1;-2;0,1)^{\mathrm{T}}$ м/с^2.

Начальное состояние (2.2) задано следующим образом:

$$r^t=(500;-600;500)^{\mathrm{T}},\ V^t=(100;100;10)^{\mathrm{T}},$$
$$r=(0,0,0)^{\mathrm{T}},\ V=(150;40;5)^{\mathrm{T}}.$$

В момент окончания функционирования системы $t_1 = 50$ должны выполняться условия (2.3):

$$x(t_1) - x^t(t_1) = 0,\ y(t_1) - y^t(t_1) = 0,\ z(t_1) - z^t(t_1) = 0\,.$$

Функционал качества управления (2.5):

$$\overline{I}(x_0, d) = \sum_{i=1}^{3} R_i^{(1)} \cdot H_i^{(1)},$$

где $R_1^{(1)} = R_2^{(1)} = R_3^{(1)} = 1$. Таким образом, необходимо решить задачу поиска оптимального терминального программного управления.

Управление, найденное с помощью метода пропорциональной навигации, и соответствующие траектории изображены на рис. 4.5.

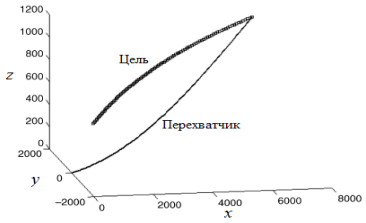


Рис. 4.5. Графики движения цели и перехватчика

Оптимальное управление, найденное с помощью обобщенного инверсного метода (при $\varepsilon = 0,001$, $\zeta = 0,1$), и соответствующие траектории («а» – для кусочно-постоянного управления, «б» – для кусочно-линейного управления) изображены на рис. 4.6, а результаты работы алгоритма при разных способах дискретизации и параметрах работы метода в табл. 4.5, 4.6 (также в эти таблицы включены результаты применения других инверсных методов).

а)

б)

Рис. 4.6. Графики движения цели и перехватчика
(обобщенный инверсный алгоритм)

Таблица 4.5. Результат работы алгоритма (кусочно-постоянное управление).

Операторы проверки и сжатия (или алгоритм)	Значение функционала качества	
	Явная схема Эйлера	Неявная схема Эйлера
FT ($w=0,01$), SAS($w=2, A=25$)	$[0;0,0943]$	$[0;0,0912]$
FTR ($w=0,01$), SAS ($w=2, A=25$)	$[0;0,0997]$	$[0;0,0973]$
FT ($w=0,01$), RPS ($A=100$)	$[0;0,0995]$	$[0;0,0996]$
FTR ($w=0,01$), RPS ($A=100$)	$[0;0,0990]$	$[0;0,0988]$
Метод дихотомии целевого интервала	$[0;0,0990]$	$[0;0,0988]$
Метод отсечки виртуальных значений	$[0;0,0990]$	$[0;0,0988]$
Метод стохастической отсечки виртуальных значений	$[0;0,0987]$	$[0;0,0988]$
Метод стохастических вырываний	$[0;0,0969]$	$[0;0,0963]$

Таблица 4.6. Результат работы алгоритма (кусочно-линейное управление).

Операторы проверки и сжатия (или алгоритм)	Значение функционала качества	
	Явная схема Эйлера	Неявная схема Эйлера
FT ($w=0,01$), SAS($w=2, A=25$)	$[0;0,0921]$	$[0;0,0893]$
FTR ($w=0,01$), SAS ($w=2, A=25$)	$[0;0,0909]$	$[0;0,0910]$
FT ($w=0,01$), RPS ($A=100$)	$[0;0,0887]$	$[0;0,0884]$
FTR ($w=0,01$), RPS ($A=100$)	$[0;0,0972]$	$[0;0,1087]$
Метод дихотомии целевого интервала	$[0;0,0880]$	$[0;0,0878]$
Метод отсечки виртуальных значений	$[0;0,0880]$	$[0;0,0878]$
Метод стохастической отсечки виртуальных значений	$[0;0,0968]$	$[0;0,0959]$
Метод стохастических вырываний	$[0;0,0968]$	$[0;0,0961]$

Параметры метода дихотомии целевого интервала: $\varepsilon=0,001$, $\zeta=0,1$.

Параметры метода отсечки виртуальных значений: $\varepsilon=0,001$, $\zeta=0,1$, $\delta=3$.

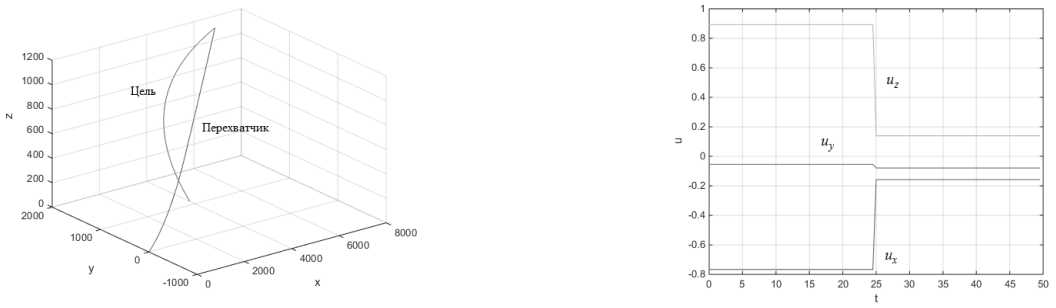
Параметры метода стохастической отсечки виртуальных значений: $\varepsilon=0,001$, $\zeta=0,1$, $\delta=3$.

Параметры метода стохастических вырываний: $\varepsilon=0,001$, $\zeta=0,1$, $A=10$, $W=5$.

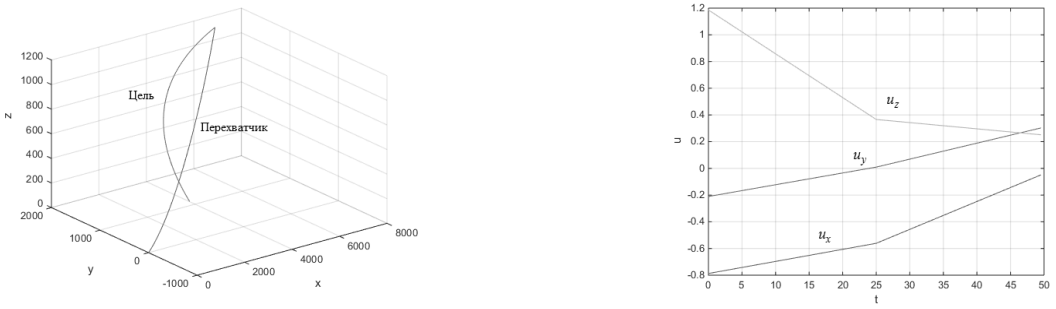
Сравнительный анализ управления и траекторий, полученных предложенным интервальным методом и методом пропорциональной навигации, свидетельствует о близости полученных результатов. Максимальное суммарное отклонение цели и

перехватчика равно $\Delta = \Delta_x + \Delta_y + \Delta_z = 0,09 + 0,14 + 0,07 = 0,3$ для кусочно-постоянного управления и $\Delta = \Delta_x + \Delta_y + \Delta_z = 0,06 + 0,08 + 0,14 = 0,28$ для кусочно-линейного управления, что соответствует требованиям, предъявляемым к точности решения задачи преследования. Здесь $\Delta_x = \max |x(t_1) - x^t(t_1)|$, $\Delta_y = \max |y(t_1) - y^t(t_1)|$, $\Delta_z = \max |z(t_1) - z^t(t_1)|$ - ошибки выполнения терминальных условий по каждой координате. Кусочно-линейное управление (табл. 4.6) предпочтительнее по величине функционала, чем кусочно-постоянное (табл. 4.5), поскольку приводит к меньшим значениям ширины интервала неопределенностей значений критерия оптимальности. Заметим, что каждому из управлений соответствуют интервалы, содержащие глобальный минимум.

Оптимальные управления, найденные с помощью метода усредненных путей, метода стохастической сетки и интервального разбросанного поиска, и соответствующие траектории движения изображены на рис. 4.7, 4.8 и 4.9 соответственно («а» – для кусочно-постоянного управления, «б» – для кусочно-линейного управления), а результаты работы алгоритмов при разных способах дискретизации сведены в табл. 4.7 – 4.10.

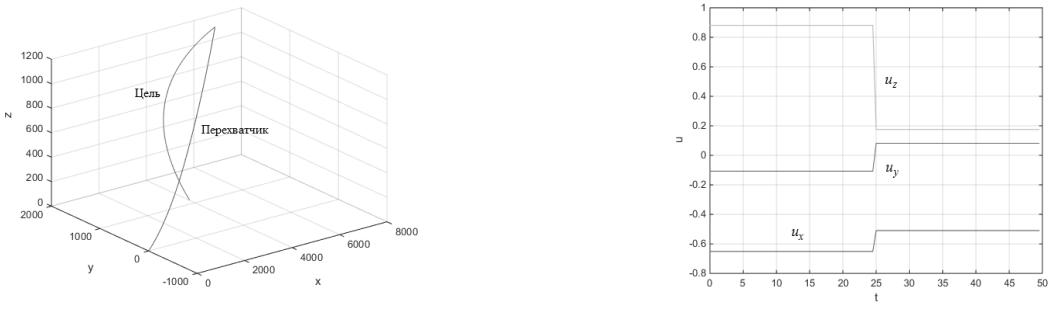


а)

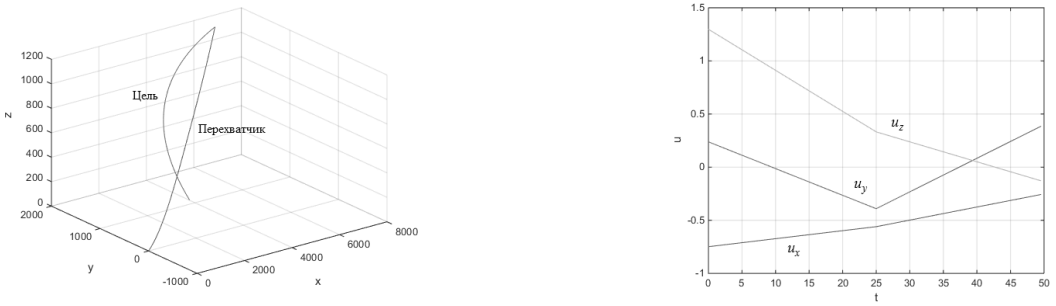


б)

Рис. 4.7. Графики движения цели и перехватчика
(метод усредненных концов путей)



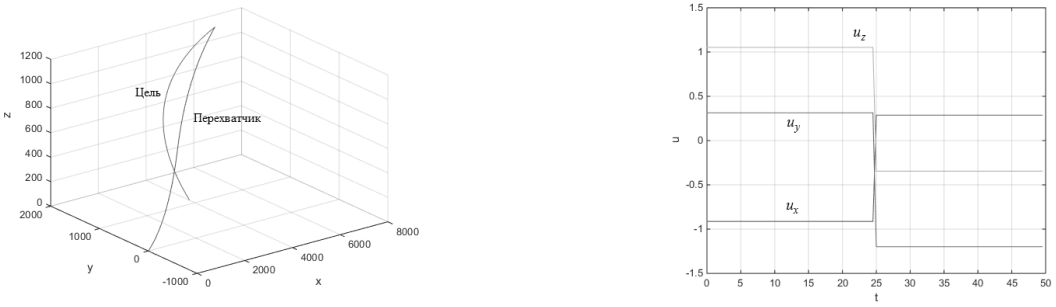
а)



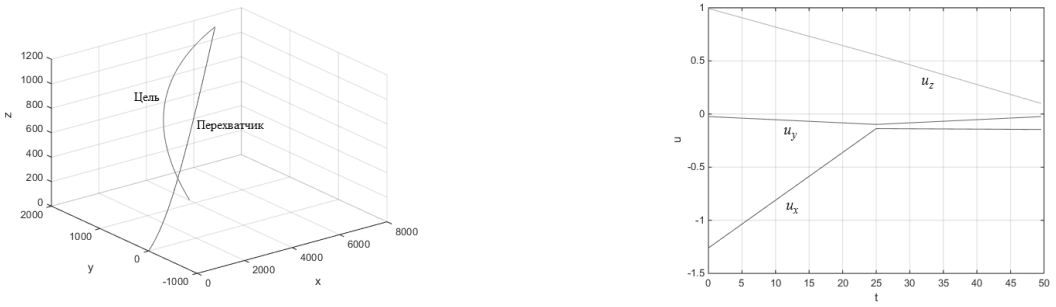
б)

Рис. 4.8. Графики движения цели и перехватчика

(метод стохастической сетки)



а)



б)

Рис. 4.9. Графики движения цели и перехватчика

(интервальный разбросанный поиск)

Параметры метода усредненных концов путей: $\varepsilon = 0{,}001, w = 0{,}5$, $p_{amount} = 75$, $p_{attempts} = 5$, $r_{\max} = 5$.

Параметры метода стохастической сетки: $\varepsilon = 0{,}001$, $w = 0{,}9$, $p_{amount} = 100$, $p_{attempts} = 100$, $p_{analyze} = 1000$.

Параметры метода интервального разбросанного поиска: $\varepsilon = 0{,}001$, $r_{gm} = 0{,}8$, $r_{rsc} = 0{,}25$, $r_{cs} = 0{,}9$, $r_{im} = 0{,}9$, $p_{size} = 100$, $rs_{size} = 10$, $sub_{size} = 2$, $\max_{im} = 1000$, $\sigma = 2500$.

Таблица 4.7. Значение функционала качества управления (кусочно-постоянное управление)

Название метода	Значение функционала качества	
	Явная схема Эйлера	Неявная схема Эйлера
Метод усредненных концов путей	$[0,0377; 0,0378]$	$[0,0371; 0,0372]$
Метод стохастической сетки	$[0,0359; 0,0360]$	$[0,0354; 0,0355]$
Метод интервального разбросанного поиска	$[0,0445; 0,0446]$	$[0,0437; 0,0438]$

Таблица 4.8. Максимальные значения отклонения цели и перехватчика (кусочно-постоянное управление)

Название метода	Значение Δ_x; Δ_y; Δ_z	
	Явная схема Эйлера	Неявная схема Эйлера
Метод усредненных концов путей	0,1040; 0,0501; 0,0401	0,1035; 0,04993; 0,0409
Метод стохастической сетки	0,0632; 0,0735; 0,0053	0,0631; 0,0733; 0,0051
Метод интервального разбросанного поиска	0,0910; 0,0912; 0,0317	0,0908; 0,091; 0,0315

Таблица 4.9. Значение функционала качества управления (кусочно-линейное управление)

Название метода	Значение функционала качества	
	Явная схема Эйлера	Неявная схема Эйлера
Метод усредненных концов путей	$[0,0347; 0,0348]$	$[0,0343; 0,0344]$
Метод стохастической сетки	$[0,0341; 0,0341]$	$[0,0340; 0,0341]$
Метод интервального разбросанного поиска	$[0,0345; 0,0346]$	$[0,0339; 0,0340]$

Таблица 4.10. Максимальные значения отклонения цели и перехватчика (кусочно-линейное управление)

Название метода	Значение Δ_x; Δ_y; Δ_z	
	Явная схема Эйлера	Неявная схема Эйлера
Метод усредненных концов путей	0,0607; 0,0698; 0,0060	0,0609; 0,0697; 0,00055
Метод стохастической сетки	0,0636; 0,0431; 0,0353	0,0632; 0,043; 0,0035
Метод интервального разбросанного поиска	0,0432; 0,0935; 0,0053	0,0431; 0,0932; 0,0051

Полученные данные о максимальных значениях отклонения являются приемлемыми с практической точки зрения и свидетельствуют об успешности применения разработанных алгоритмов. Кусочно-линейное управление (табл. 4.9, 4.10) предпочтительнее по величине функционала, чем кусочно-постоянное (табл. 4.7, 4.8).

4.2.2. ЗАДАЧА ОБ УПРАВЛЕНИИ СОЛНЕЧНЫМ ПАРУСОМ

Рассматривается задача поиска оптимального по быстродействию терминального программного управления перспективным космическим летательным аппаратом (КЛА) – солнечным парусом, реализующим межпланетную миссию [122]. Данная задача заключается в переходе с орбиты Земли на орбиту Меркурия за минимальное время.

Система (2.1), описывающая движение космического летательного аппарата, выглядит следующим образом:

$$\dot{r}(t) = u,\ \dot{\theta}(t) = v / r,$$
$$\dot{u}(t) = v^2 / r - (\mu / r^2) \cdot (1 - \beta \cdot \cos^3 \alpha),$$
$$\dot{v}(t) = -u \cdot v / r + \mu \cdot \beta \cdot \sin \alpha \cdot \cos^2 \alpha / r^2,$$

где r, θ – радиальная и угловая позиции соответственно, u, v – радиальная и тангенциальная скорости, α – угол тангажа (переменная управления), $\beta = 0,042$ – параметр яркости солнечного паруса, $\mu = G \cdot M_S \approx 1,327474512 \cdot 10^{20}$ м$^3 \cdot$с$^{-2}$ – солнечное гравитационное ускорение, $G = 6,67408 \cdot 10^{-11}$ м$^3 \cdot$кг$^{-1} \cdot$с$^{-2}$ – универсальная гравитационная постоянная, $M_S = 1,989 \cdot 10^{30}$ кг – масса Солнца. Ограничение на управление: $\boldsymbol{U} = [-\pi / 2; \pi / 2]$.

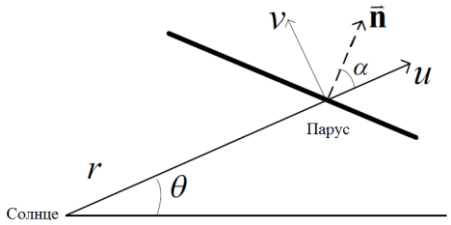


Рис. 4.10. Принятая система координат

Начальное состояние (2.2) задано следующим образом:

$$t_0 = 0,\ r(t_0) = 1 \text{ AU} = 1,496 \cdot 10^{11} \text{ м},\ \theta(t_0) = 0,$$
$$u(t_0) = 0 \text{ км/с} = 0 \text{ м/с},\ v(t_0) = 29,8 \text{ км/с} = 2,98 \cdot 10^4 \text{ м/с}.$$

В момент окончания функционирования системы должны выполняться условия (2.3):

$$r(t_1) - 5,8344 \cdot 10^{10} = 0,$$
$$u(t_1) = 0,$$
$$v(t_1) - 4,79 \cdot 10^4 = 0.$$

Функционал качества управления (2.5):

$$\bar{I}(x_0, d) = \underbrace{\frac{t_1}{86400}}_{I(d)} + \sum_{i=1}^{3} R_i^{(1)} \cdot H_i^{(1)},$$

где $R_1^{(1)} = R_2^{(1)} = R_3^{(1)} = 10^5$.Таким образом, необходимо решить задачу поиска оптимального программного терминального управления, оптимального по быстродействию.

На рис. 4.11 изображены графики траекторий солнечного паруса и управлений, которые были найдены в [122].

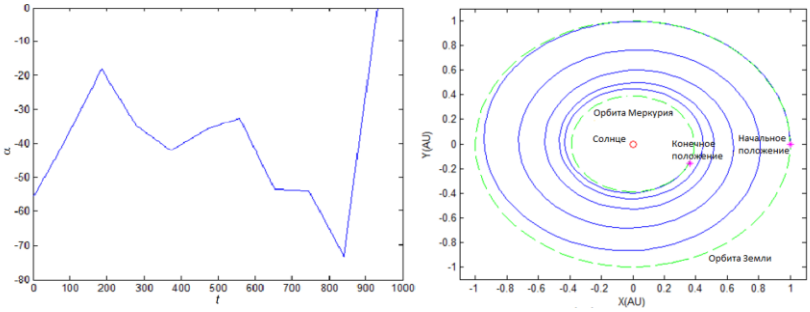
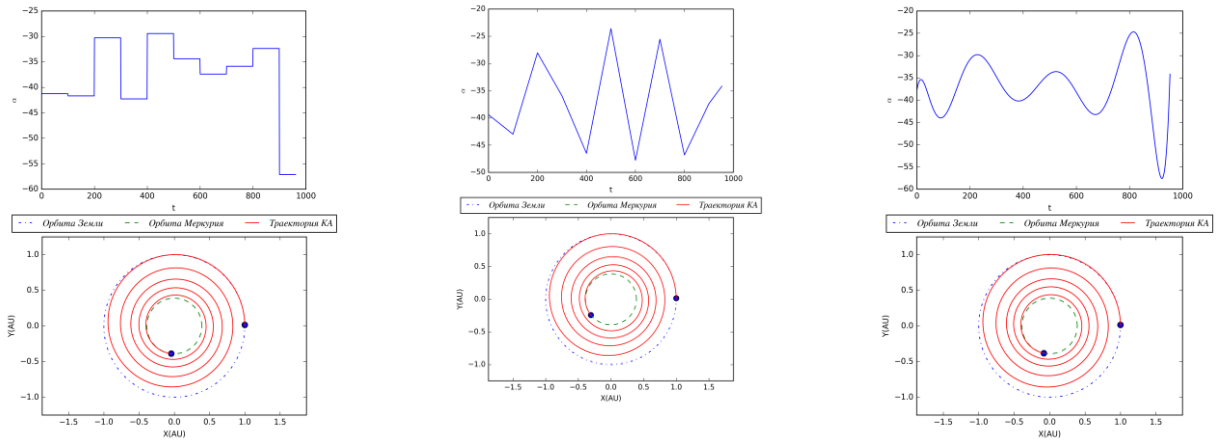


Рис. 4.11. Управление и соответствующая траектория солнечного паруса ($I^* = 933,9$)

На рис. 4.12 изображены управления различных типов (кусочно-постоянное, кусочно-линейное и в виде разложения по системе полиномов Лежандра), найденные с помощью интервального метода взрывов ($t^g_{\max} = t^c_{\max} = 1000, B_{\max} = 100, \dot{r}_{\max} = 0,01$), и соответствующие им траектории.



а) б) в)

Рис. 4.12. Управления, найденные в разных классах функций, и траектории

С помощью предложенной в разделе 2.1 методики найдены программные управления, принадлежащие к разным классам функций (переход с орбиты Земли на орбиту Меркурия для кусочно-постоянного управления был произведен за 961 день, для кусочно-линейного – за 953 дня, для управления, найденного в виде разложения по базису, – за 952 дня; использовались интервальный метод взрывов, интервальный генетический алгоритм и

адаптивный интервальный алгоритм соответственно). В работах Wang Y., Zhu M., Wei Y., McInnes C.R., Hughes G.W. были найдены управления, с помощью которых этап перехода на орбиту Меркурия завершался за 933,9 [122], 1210,0 [103] и 1043,3 [95] дней соответственно.

4.2.3. ЗАДАЧА О КОМАНДНОЙ НАВИГАЦИИ

В [98] рассматривается задача командной навигации. Ее смысл заключается в синхронном приведении группы объектов в заданное конечное состояние.

Система (2.1), описывающая движение группы объектов выглядит следующим образом:

$$\dot{x}_i(t) = V_i \cdot \cos\gamma_i,\ \dot{y}_i(t) = V_i \cdot \sin\gamma_i,\ \dot{\gamma}_i(t) = u_i / V_i,\ i = \overline{1,G}$$

где x_i, y_i описывают положение объекта, γ_i – направление, V_i – скорость, u_i – управление, G – количество объектов в группе. Ограничение на управление: $\boldsymbol{U} = \underbrace{[-50;50]\times\ldots\times[-50;50]}_{G}$.

Рассмотрим несколько сценариев для данной задачи.

Начальное состояние (2.2) задано следующим образом:

а) $G = 2, t_0 = 0,$ $\begin{cases} x_1(t_0) = 1180 \text{ м}, y_1(t_0) = 2080 \text{ м}, \gamma_1(t_0) = -20^\circ, V_1 = 290 \text{ м/с}, \\ x_2(t_0) = 0 \text{ м}, y_2(t_0) = 0 \text{ м}, \gamma_2(t_0) = -5^\circ, V_2 = 300 \text{ м/с}. \end{cases}$

б) $G = 4, t_0 = 0,$ $\begin{cases} x_1(t_0) = -8000 \text{ м}, y_1(t_0) = 0 \text{ м}, \gamma_1(t_0) = 0^\circ, V_1 = 350 \text{ м/с}, \\ x_2(t_0) = 0 \text{ м}, y_2(t_0) = 12000 \text{ м}, \gamma_2(t_0) = -90^\circ, V_2 = 200 \text{ м/с}, \\ x_3(t_0) = 1000 \text{ м}, y_3(t_0) = 0 \text{ м}, \gamma_3(t_0) = 180^\circ, V_3 = 250 \text{ м/с}, \\ x_4(t_0) = 0 \text{ м}, y_4(t_0) = -6000 \text{ м}, \gamma_4(t_0) = 90^\circ, V_4 = 200 \text{ м/с}. \end{cases}$

В момент окончания функционирования системы должны выполняться условия (2.3):

а) $x_i(t_1) - 13000 = 0, y_i(t_1) = 0, i = \overline{1,G};$

б) $x_i(t_1) = 0, y_i(t_1) = 0, i = \overline{1,G}.$

Функционал качества управления (2.5):

$$\bar{I}(x_0, d) = t_1 + \sum_{i=1}^{2\cdot G} R_i^{(1)} \cdot H_i^{(1)},$$

где $R_i^{(1)} = 10^2, i = 1,...,2\cdot G$. Таким образом, необходимо решить задачу поиска оптимального по быстродействию программного терминального управления.

На рис. 4.13 изображены графики траекторий и управлений, которые были найдены в [98] с помощью алгоритмов пропорциональной и командной пропорциональной навигации (PN и CPN соответственно), а на рис. 4.14 – с помощью интервального метода взрывов ($t^g_{\max} = t^c_{\max} = 2500, B_{\max} = 100, r^{\cdot}_{\max} = 1$).

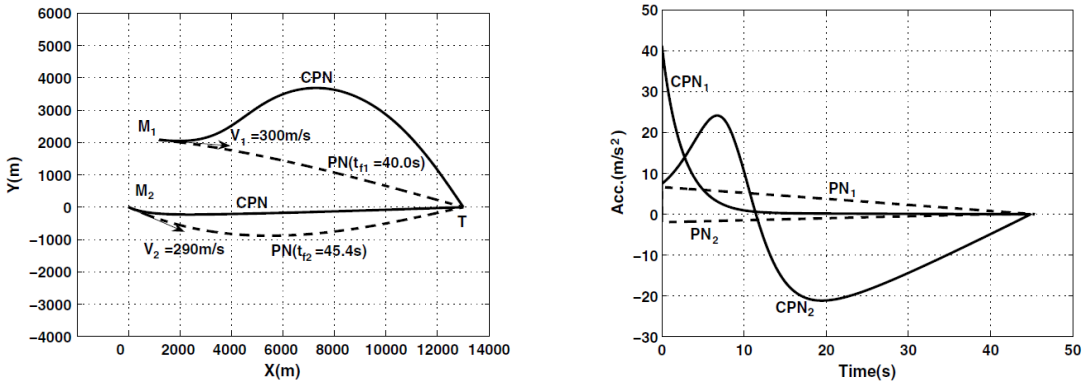


Рис. 4.13. Траектории и управление, найденные в [98]

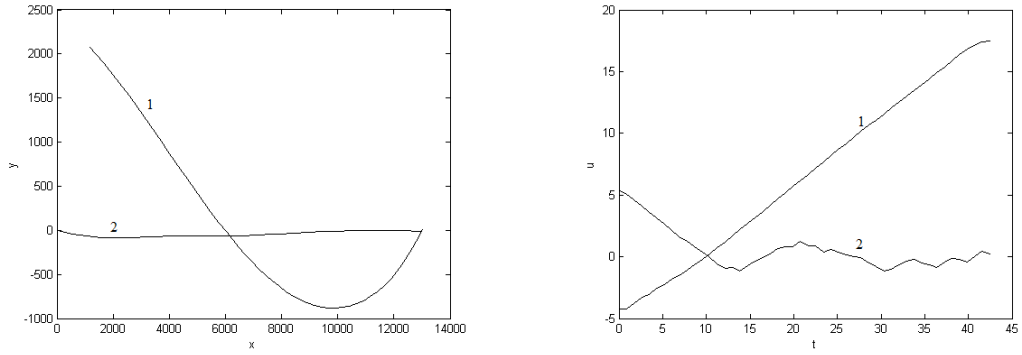


Рис. 4.14. Траектории и управления, найденные с помощью интервального метода взрывов

С помощью интервального метода взрывов получилось не только привести объекты в заданное состояние, но и сделать это за меньшее время ~ 43 с (на почти две секунды быстрее). Суммарное отклонение по целям составило $[0;10,1]$.

Результаты, полученные для четырех целей (время наведения $\sim 59,5$ с, суммарное отклонение по целям составило $[0;19,2]$), представлены на рис. 4.15.

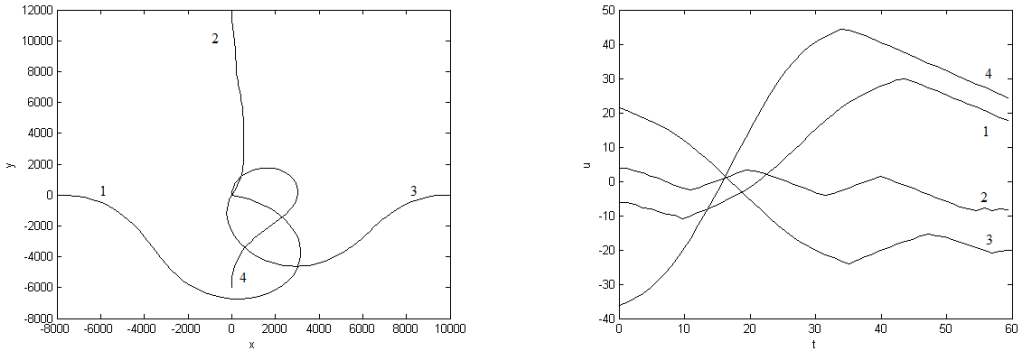


Рис. 4.15. Траектории и управления, найденные с помощью интервального метода взрывов

Таким образом, алгоритм успешно справился с поставленной задачей и подтвердил свою эффективность.

4.2.4. ЗАДАЧА О ПРИЗЕМЛЕНИИ ГИПЕРЗВУКОВОГО ЛЕТАТЕЛЬНОГО АППАРАТА

Система дифференциальных уравнений (2.1), описывающих динамику ГЗЛА при его приземлении, выглядит следующим образом [118]:

$$\begin{cases} \dot{r}(t) = V \cdot \sin\gamma, \\ \dot{\theta}(t) = \dfrac{V\cos\gamma\cos\psi}{r\cos\phi}, \\ \dot{\phi}(t) = \dfrac{V\cos\gamma\sin\psi}{r}, \\ \dot{V}(t) = \dfrac{-D}{m} - g\sin\gamma + \Omega^2 r\cos\phi(\sin\gamma\cos\phi - \cos\gamma\sin\phi\sin\psi), \\ \dot{\gamma}(t) = \dfrac{L\cos\sigma}{mV} + (\dfrac{V}{r} - \dfrac{g}{V})\cos\gamma + 2\Omega\cos\phi\cos\psi + \dfrac{\Omega^2 r}{V}\cos\phi(\cos\gamma\cos\phi + \sin\gamma\sin\phi\sin\psi), \\ \dot{\psi}(t) = \dfrac{L\sin\sigma}{mV\cos\gamma} - \dfrac{V}{r}\cos\gamma\cos\psi\tan\phi + 2\Omega(\tan\gamma\cos\phi\sin\psi - \sin\phi) - \dfrac{\Omega^2 r}{V\cos\gamma}\sin\phi\cos\phi\cos\psi, \end{cases}$$

где r - расстояние от центра Земли, θ - долгота, ϕ - широта, V - скорость относительно Земли, γ - угол наклона траектории полета, ψ - угол азимута скорости, σ - угол крена, $g = \dfrac{GM}{r^2}$ - ускорение, GM - гравитационная постоянная Земли, Ω - скорость вращения Земли, m - масса ГЗЛА, $L = \dfrac{1}{2}\rho_0 e^{-\beta(r-r_0)}V^2 C_L S_{ref}$ - аэродинамическая подъемная сила, $D = \dfrac{1}{2}\rho_0 e^{-\beta(r-r_0)}V^2 C_D S_{ref}$ - аэродинамическая сила тяги, ρ_0 - плотность воздуха на уровне моря, β - коэффициент, r_0 - константа, C_L, C_D - коэффициенты аэродинамических сил подъема и тяги, S_{ref} - относительная площадь поверхности ГЗЛА.

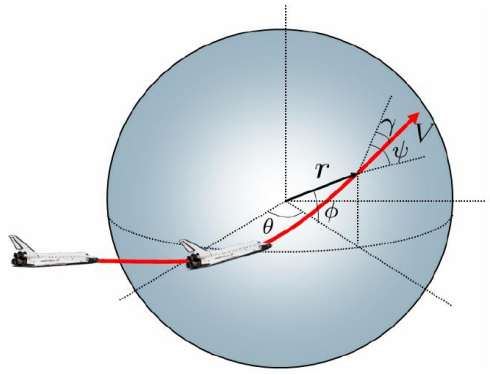


Рис. 4.16. Траектория движения ГЗЛА

Коэффициенты аэродинамических сил подъема и тяги берутся как полиномиальная функция от числа Маха $M = V / V_s$ (V_s – скорость звука) и угла атаки α на основе информации об аппарате X-33:

$$C_L = -0.0005225\alpha^2 + 0.03506\alpha - 0.04857M + 0.1577,$$
$$C_D = 0.0001432\alpha^2 + 0.00558\alpha - 0.01048M + 0.2204.$$

Числовые параметры, используемые в модели: $m = 35828$ кг, $S_{ref} = 149,3881$ м^2, $\rho_0 = 1,2256$ кг/м^3, $r_0 = 6,3712 \cdot 10^6$ м, $\beta = 1,3785 \cdot 10^{-4}$ м$^{-1}$, $GM = 3,986 \cdot 10^{14}$ м^3/с^2, $\Omega = 7,2722 \cdot 10^{-5}$ рад/с, $R_e = 6,3781 \cdot 10^6$ м, $V_e = 11180$ м/с.

Вектор состояния состоит из 6 компонент: $x(t) = (r(t), \theta(t), \phi(t), V(t), \gamma(t), \psi(t))^{\mathrm{T}}$. Управление ГЗЛА осуществляется за счет углов атаки и крена: $u(t) = (\alpha(t), \sigma(t))^{\mathrm{T}}$.

Начальное состояние (2.2) системы: $r(t_0) = 6429100$, $\theta(t_0) = -85$, $\phi(t_0) = 30$, $V(t_0) = 2600$, $\gamma(t_0) = -1.3$, $\psi(t_0) = 0$.

Ограничения на терминальное состояние (2.3):

$$6378578 \le r(t_1) \le 6378822, \qquad V(t_1) = 91,44,$$
$$-80,7126 \le \theta(t_1) \le -80,7098, \qquad \gamma(t_1) = -6,$$
$$29,998903 \le \phi(t_1) \le 30,001097, \qquad \psi(t_1) = -60.$$

Фазовые ограничения на траекторию ГЗЛА:

$$q(t) \in [0;14364,1], \quad h(t) = r(t) - R_e \in [0;121920], \quad V(t) \in [0,3;5200],$$
$$Q(t) \in [0;242000], \quad \theta(t) \in [-90;90], \quad \gamma(t) \in [-89;89],$$
$$n_z^g(t) = n_z(t)/g \in [-2,5;2,5], \quad \phi(t) \in [-89;89], \quad \psi(t) \in [-180;180],$$

где $q = \frac{1}{2}\rho_0 e^{-\beta(r-r_0)}V^2$ - динамическое давление, $Q = 4.477228 \cdot 10^{-9}\sqrt{\rho_0 e^{-\beta(r-r_0)}}V^{3.15}$ - скорость нагрева, $n_z = \frac{|L\cos\alpha + D\sin\alpha|}{m}$ - нормальное ускорение.

Ограничения на скорость изменения вектора управления:

$$\dot{\alpha}(t) \in [-5;5] \Leftrightarrow \frac{\alpha(\tau_{i+1}) - \alpha(\tau_i)}{\tau_{i+1} - \tau_i} \in [-5;5],$$
$$\dot{\sigma}(t) \in [-5;5] \Leftrightarrow \frac{\sigma(\tau_{i+1}) - \sigma(\tau_i)}{\tau_{i+1} - \tau_i} \in [-5;5].$$

Функционал качества управления (2.5):

$$\bar{I}(x_0, d) = \sum_{i=1}^{6} R_i^{(1)} \cdot H_i^{(1)} + \sum_{j=1}^{11} R_j^{(2)} \cdot H_j^{(2)},$$

где $R_1^{(1)} = 1000$, $R_2^{(1)} = R_3^{(1)} = R_5^{(1)} = R_6^{(1)} = 10^6$, $R_4^{(1)} = 10^4$ и $R_1^{(2)} = R_4^{(2)} = R_7^{(2)} = R_8^{(2)} = 10$, $R_2^{(2)} = R_3^{(2)} = R_5^{(2)} = R_6^{(2)} = R_9^{(2)} = 10^4$, $R_{10}^{(2)} = R_{11}^{(2)} = 1$.Таким образом, необходимо решить задачу поиска оптимального программного терминального управления.

Графики на рис. 4.17 демонстрируют результаты из [118], полученные с помощью B-сплайнов и метода Галеркина.

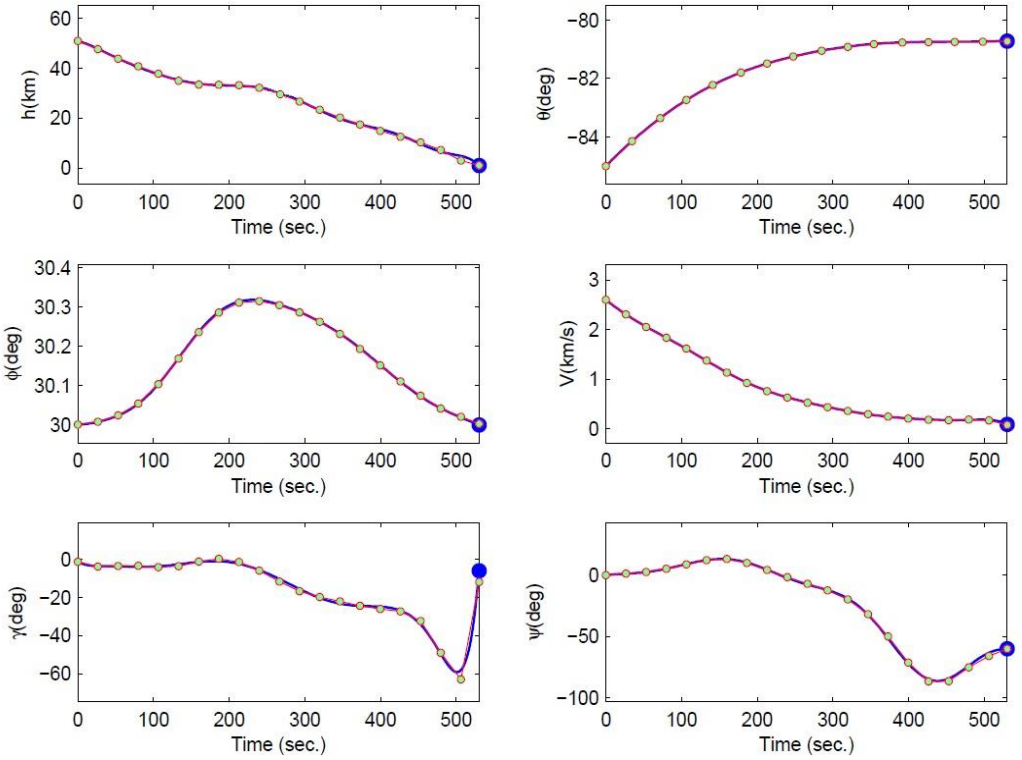


Рис. 4.17. Компоненты вектора состояния

Графики на рис. 4.18 и 4.20 иллюстрируют результаты, полученные с помощью интервального метода взрывов ($t^g_{\max} = t^c_{\max} = 5000, B_{\max} = 100, \dot{r}_{\max} = 0{,}1$), для кусочно-постоянного и кусочно-линейного управлений соответственно.

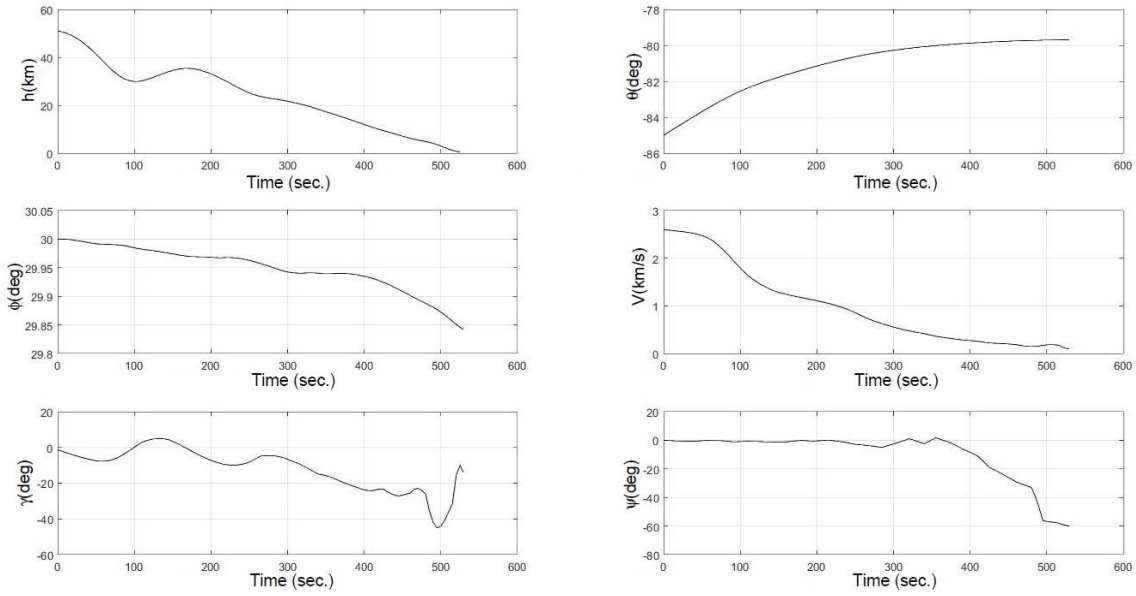


Рис. 4.18. Компоненты вектора состояния (кусочно-постоянное управление)

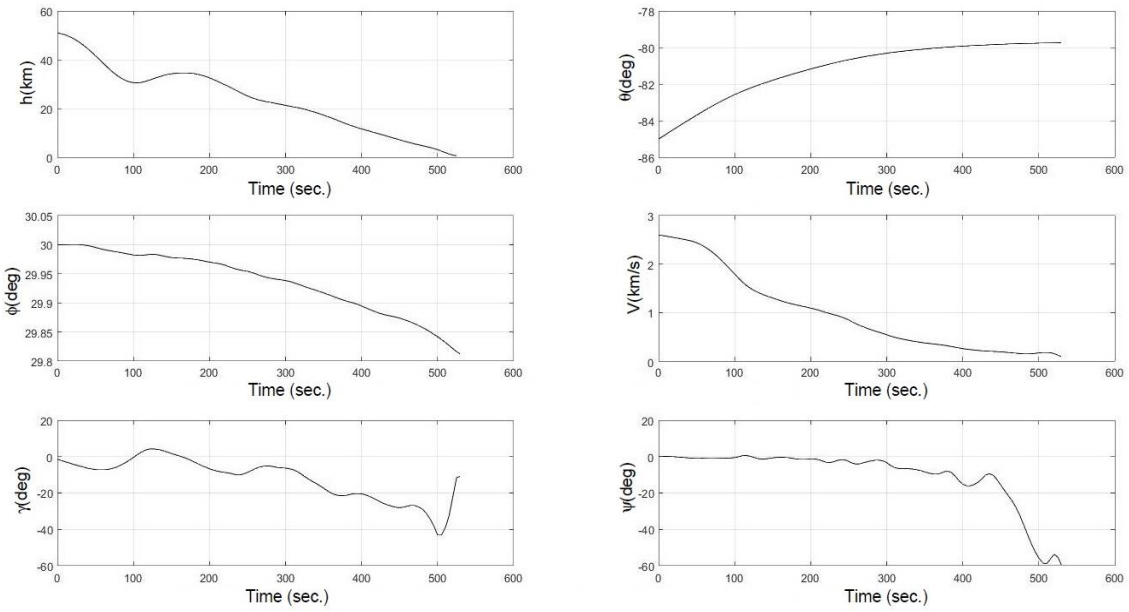


Рис. 4.19. Компоненты вектора состояния (кусочно-линейное управление)

Информация по выполнению терминальных ограничений представлена в табл. 4.11, 4.12

Табл. 4.11. Терминальные ограничения (кусочно-постоянное управление)

$r(t_1)$	$[6378577,87; 6378579,23]$	$\gamma(t_1)$	$[-14,44; -14,41]$
$V(t_1)$	$[91,43; 91,47]$	$\phi(t_1)$	$[29,84; 29,99]$
$\theta(t_1)$	$[-80,12; -79,69]$	$\psi(t_1)$	$[-60,10; -59,98]$

Табл. 4.12. Терминальные ограничения (кусочно-линейное управление)

$r(t_1)$	$[6378578,12; 6378579,52]$	$\gamma(t_1)$	$[-9,76; -9,65]$
$V(t_1)$	$[91,36; 91,41]$	$\phi(t_1)$	$[29,81; 29,93]$
$\theta(t_1)$	$[-80,73; -80,72]$	$\psi(t_1)$	$[-59,98; -59,88]$

Очевидно, что лучше использовать кусочно-линейное управление для более точного выполнения ограничений. Все ограничения были выполнены с достаточной точностью.

Таким образом, путем сравнения результатов можно сделать вывод, что интервальные алгоритмы глобальной условной оптимизации могут с достаточно высокой эффективностью быть применены для решения задачи оптимального управления ГЗЛА.

4.2.5. ЗАДАЧА О СТАБИЛИЗАЦИИ СПУТНИКА

Рассматривается задача гашения вращательного движения спутника с помощью установленных на нем двигателей [11]. Система (2.1), описывающая движение твердого тела относительно центра инерции после перехода к безразмерным переменным, имеет вид:

$$\dot{p}(t) = u_1 / 6,$$
$$\dot{q}(t) = u_2 - 0,2 \cdot r \cdot p,$$
$$\dot{r}(t) = 0,2 \cdot (u_3 + p \cdot q),$$

где p, q, r – проекции угловой скорости на главные центральные оси инерции, а u_1, u_2, u_3 – управления, которые характеризуют тяги двигателей, расположенных на спутнике. Ограничение на управление: $\boldsymbol{U} = [-200; 200] \times [-200; 200] \times [-200; 200]$.

Множество начальных состояний (2.7) задано следующим образом:

$$t_0 = 0,\ \boldsymbol{\Omega} = [23; 25] \times [13; 15] \times [13; 15].$$

В момент окончания функционирования системы $t_1 = 1$ должны выполняться условия (2.8):

$$p(t_1) = q(t_1) = r(t_1) = 0.$$

Функционал качества управления (2.9):

$$I_i(x_0, d) = \int_{t_0}^{t_1} |u_1(t)| + |u_2(t)| + |u_3(t)|\, dt.$$

Преобразуем функционал качества управления следующим образом:

$$\overline{I}(x_0, d) = I_i(x_0, d) + \underbrace{\sum_{i=1}^{3} R_i^{(1)} \cdot H_i^{(1)}}_{I_t},$$

где $R_1^{(1)} = R_2^{(1)} = R_3^{(1)} = 10^3$ – параметры штрафа.

Результаты, полученные в [11] для фиксированного начального условия $p(t_0) = 24, q(t_0) = r(t_0) = 16$, изображены на рис. 4.20. Значение функционала при этом составило $I^* \approx 169,42$.

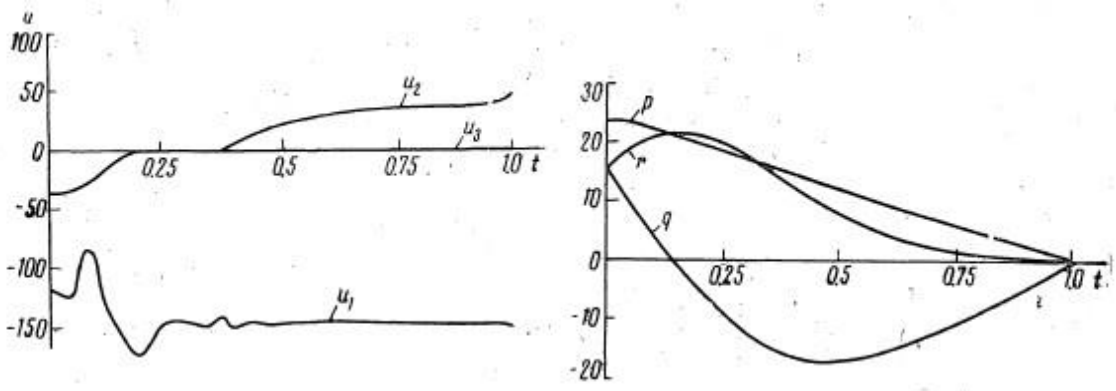


Рис. 4.20. Программное управление и соответствующие ему траектории из [11]

Используя самоорганизующийся интервальный алгоритм имитации эволюции колонии бактерий ($N_{\max} = 5000, w_\varepsilon = 0,00001$), найдем кусочно-постоянное программное управление для тех же начальных условий. Полученное управление изображено на рис. 4.21. Значение функционала при этом составило $[172,0956; 172,0973]$.

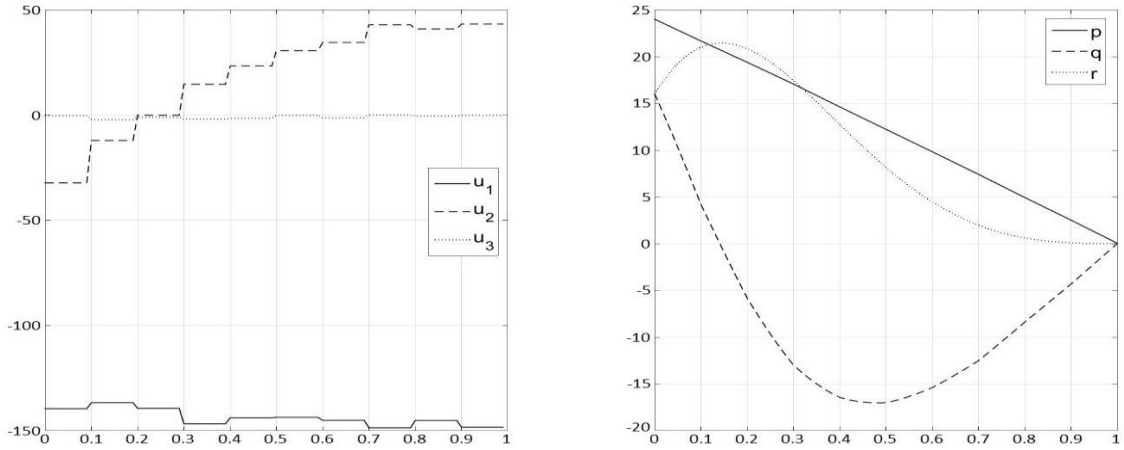


Рис. 4.21. Программное управление и соответствующие ему траектории при использовании самоорганизующегося интервального алгоритма имитации эволюции колонии бактерий

В результате решения задачи был найден наилучший брус $\boldsymbol{a}$, имеющий структуру (2.17):

$$\boldsymbol{a}^* = \begin{pmatrix} \begin{pmatrix} [-3,9431;-3,9430] \\ [-1,1209;-1,1208] \\ [-4,3391;-4,3990] \\ [87,3999;87,4001] \end{pmatrix}^{\mathrm{T}} & \begin{pmatrix} [4,1333;4,1335] \\ [-9,9832;-9,9830] \\ [-2,4021;-2,4019] \\ [77,3211;77,3212] \end{pmatrix}^{\mathrm{T}} & \cdots \end{pmatrix}^{\mathrm{T}}$$

Для синтеза управления с неполной обратной связью используем полную систему ортонормированных полиномов Лежандра. Пример полученного управления и соответствующий пучок траекторий изображены на рис. 4.22.

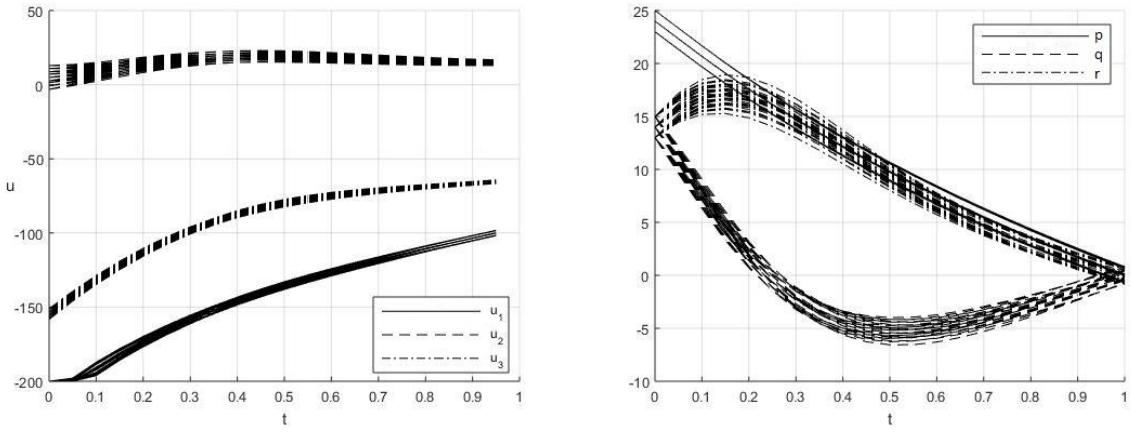


Рис. 4.22. Траектория и управления (при $x^1 = (p,q)^T$)

В таблице 4.13 приведены результаты применения интервальных алгоритмов оптимизации для различных конфигураций вектора x^1, по которому имеется обратная связь.

Табл. 4.13. Значения компонент функционала качества управления

Алгоритм	$u(t,x^1)$	$\boldsymbol{I}_i$	$\boldsymbol{I}_t$	$\boldsymbol{I}$
AIA	$x^1 = p$	[200,55; 200,78]	[1,02; 1,02]	[1220,55; 1220,78]
IGA	$x^1 = q$	[199,31; 200,01]	[1,01; 1,01]	[1209,31; 1210,01]
IES	$x^1 = r$	[201,94; 202,09]	[1,07; 1,07]	[1272,09; 1272,94]
AIA	$x^1 = (p,q)^T$	[182,13; 182,93]	[0,52; 0,52]	[702,13; 702,93]
IGA	$x^1 = (p,r)^T$	[179,20; 179,82]	[0,46; 0,46]	[639,20; 639,82]
IES	$x^1 = (q,r)^T$	[181,69; 181,99]	[0,49; 0,49]	[671,69; 671,99]
AIA	$x^1 = (p,q,r)^T$	[173,09; 174,02]	[0,14; 0,14]	[313,09; 314,02]

Параметры адаптивного интервального алгоритма: $N_{\max} = 2500, \gamma = 0,98, w_{init} = 0,8$, $N_B = 8, \gamma_B = 0,98, C_B^+ = C_B^- = 4, \alpha_B = \beta_B = 0,4$, $r^B = 1$, $q^+ = q^- = (0,4; 0,3; 0,2; 0,1)^{\mathrm{T}}$, $N_W = 5, Sp = (10;10)^{\mathrm{T}}$, $\rho_W = 0,2, \gamma_W = 0,9$, $A_{good} = 3$, $A_{bad} = 2$, $N_A = 10, \gamma_A = 0,98, A_{\max} = 10, r^A = 1$.

Параметры интервального генетического алгоритма: $t_{\max} = 1000, \varepsilon = 0,001, pop_{amount} = 100, pool_{amount} = 20, r = 0,025$, LR кодирование, одноточечные скрещивание и мутация, рулетка, простое усечение.

Параметры интервального метода взрывов: $t^g_{\max} = t^c_{\max} = 2500, B_{\max} = 100, r_{\max} = 1$.

По полученным данным видно, что задача была решена успешно.

4.2.6. ЗАДАЧА О ПЕРЕХВАТЕ

Система (2.1), описывающая движения цели (T) и перехватчика (I), выглядит следующим образом [121]:

$$\dot r_T(t) = V_T \cdot \cos(\gamma_T - \lambda_T), \qquad r_I(t) = V_I \cdot \cos(\gamma_I - \lambda_I),$$
$$\dot\lambda_T(t) = V_T \cdot \sin(\gamma_T - \lambda_T)/r_T, \qquad \dot\lambda_I(t) = V_I \cdot \sin(\gamma_I - \lambda_I)/r_I,$$
$$\dot\gamma_T(t) = u_T/V_T, \qquad \dot\gamma_I(t) = u_I/V_I,$$

где V_T, V_I – скорости, r_T, r_I – расстояния от начала координат до цели и до перехватчика, λ_T, λ_I – угол линии визирования, γ_T, γ_I – путевые углы, θ, δ – углы отклонения скорости, $u_T = -10, u_I$ – поперечные ускорения (u_I – переменная управления в задаче) (рис. 4.14). Ограничение на управление: $\boldsymbol{U} = [-70; 70]$.

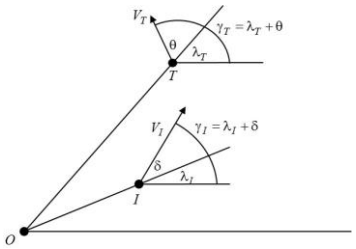


Рис. 4.23. Кинематика движения цели и перехватчика

В момент окончания функционирования системы должны выполняться условия (2.21), соответствующие равенству координат цели и перехватчика:

$$\begin{matrix} r_T(t_1)\cdot\cos\lambda_T(t_1)-r_I(t_1)\cdot\cos\lambda_I(t_1)=0, \\ r_T(t_1)\cdot\sin\lambda_T(t_1)-r_I(t_1)\cdot\sin\lambda_I(t_1)=0. \end{matrix}.$$

Вследствие того, что данная задача является задачей поиска терминального управления, оптимального по быстродействию, то функционал качества управления (2.23) имеет вид:

$$I(x_0, p^1, p^2, d) = t_1.$$

Преобразуем функционал качества управления следующим образом:

$$\overline{I}(x_0, p^1, p^2, d) = I(x_0, p^1, p^2, d) + \sum_{i=1}^{2} R_i^{(1)} \cdot H_i^{(1)},$$

где $R_1^{(1)} = R_2^{(1)} = 10^3$ – параметры штрафа.

Определим множество возможных значений характерных параметров объекта (2.22):

$$p^1 = (V_T, V_I)^T \in \boldsymbol{p}^1 = [100;150]\times[175;225].$$

Уравнение модели измерений (2.22) запишем в следующей форме:

$$\begin{cases} z_1 = r_T - r_I, \\ z_2 = \lambda_T - \lambda_I, \\ z_3 = \gamma_I. \end{cases}$$

Таким образом, цель задачи – найти оптимальное по быстродействию терминальное управление по выходу в виде разложения по системе ортонормированных базисных функций.

Рассмотрим три случая задания множества возможных начальных состояний (2.20):

а) разное начальное положение: $\boldsymbol{\Omega}^a = [1750;2250]\times\pi/4\times\pi/2\times[750;1250]\times\pi/4\times\pi/4$,

б) разный угол линии визирования: $\boldsymbol{\Omega}^б = 2000\times[7\cdot\pi/36;11\cdot\pi/36]\times\pi/2\times1000\times$ $\times[7\cdot\pi/36;11\cdot\pi/36]\times\pi/4$,

в) разный путевой угол: $\boldsymbol{\Omega}^в = 2000\times\pi/4\times[7\cdot\pi/18;11\cdot\pi/18]\times1000\times\pi/4\times[7\cdot\pi/36;11\cdot\pi/36]$.

Для всех сценариев приняты масштабы усечения: $A_0 = A_1 = A_2 = A_3 = 4$.

В результате решения задачи был найден наилучший брус $\boldsymbol{a}$, имеющий структуру (2.32):

$$\boldsymbol{a}^* = \left(\begin{pmatrix} [-166,229;-166,228] \\ [-54,368;-54,367] \\ [-147,905;-147,904] \\ [-96,603;-96,602] \end{pmatrix}^{\mathrm{T}} \quad \begin{pmatrix} [-48,457;-48,456] \\ [91,302;91,303] \\ [-7,559;-7,558] \\ [52,965;52,966] \end{pmatrix}^{\mathrm{T}} \quad \ldots \right)^{\mathrm{T}}$$

На рис. 4.24 изображены графики законов управления и соответствующие траектории (сплошной линией обозначается движение цели, пунктирной – перехватчика) для трех описанных сценариев. В табл. 4.14 приведены границы интервала значений функционала (2.33): $\tilde{J}(\boldsymbol{a})$, а также величина промаха, определяемая формулой

$$\Delta = \max_{j_\Omega, j_p, j_q} \left\| \begin{pmatrix} r_T^{j_\Omega, j_p, j_q}(t_1)\ \cos\lambda_T^{j_\Omega, j_p, j_q}(t_1) \\ r_T^{j_\Omega, j_p, j_q}(t_1)\ \sin\lambda_T^{j_\Omega, j_p, j_q}(t_1) \end{pmatrix} - \begin{pmatrix} r_I^{j_\Omega, j_p, j_q}(t_1)\ \cos\lambda_I^{j_\Omega, j_p, j_q}(t_1) \\ r_I^{j_\Omega, j_p, j_q}(t_1)\ \sin\lambda_I^{j_\Omega, j_p, j_q}(t_1) \end{pmatrix} \right\|_E$$

где $\|\cdot\|_E$ - Евклидова норма.

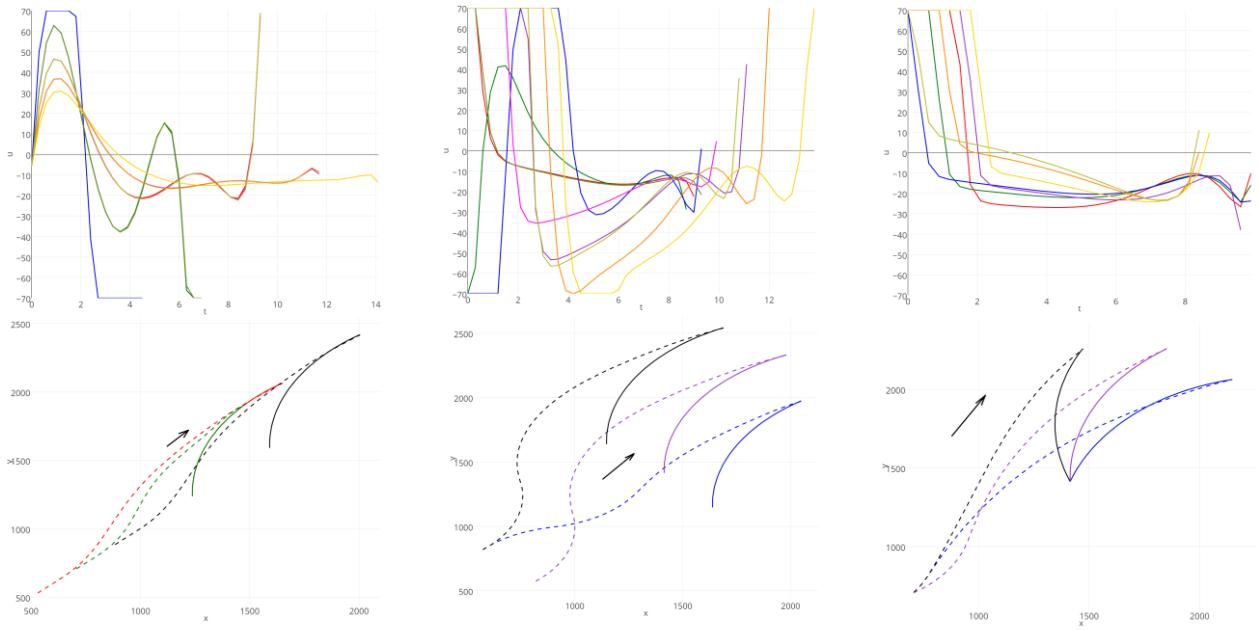
Таблица 4.14. Описание сценариев при неполной информации о параметрах объекта

Сценарий	а)	б)	в)
Метод оптимизации	IGA	AIA	IES
Базисная система	Полиномы Лежандра	Полиномы Чебышева	Полиномы Гегенбауэра ($\alpha = 1$)
Множество начальных состояний	$\boldsymbol{\Omega}^a$	$\boldsymbol{\Omega}^б$	$\boldsymbol{\Omega}^в$
Значения функционала качества управления	[3782.73;3783.88]	[3307.61;3309.03]	[3815.53;3816.91]
Максимальный промах	$\Delta = 6.73$	$\Delta = 5.073$	$\Delta = 5.323$

Параметры адаптивного интервального алгоритма (AIA): $N_{\max} = 2500, \gamma = 0,98, w_{init} = 0,8$, $N_B = 8, \gamma_B = 0,98, C_B^+ = C_B^- = 4, \alpha_B = \beta_B = 0,4$, $r^B = 1$, $q^+ = q^- = (0,4;0,3;0,2;0,1)^{\mathrm{T}}$, $N_W = 5, Sp = (10;10)^{\mathrm{T}}$, $\rho_W = 0,2, \gamma_W = 0,9$, $A_{good} = 3$, $A_{bad} = 2$, $N_A = 10, \gamma_A = 0,98, A_{\max} = 10, r^A = 1$.

Параметры интервального генетического алгоритма (IGA): $t_{\max} = 1000, \varepsilon = 0,001, pop_{amount} = 100, pool_{amount} = 20, r = 0,025$, LR кодирование, одноточечные скрещивание и мутация, рулетка, простое усечение.

Параметры интервального метода взрывов (IES): $t_{\max}^g = t_{\max}^c = 2500, B_{\max} = 100, r_{\max} = 1$.



а) б) в)

Рис. 4.24. Управления и соответствующие траектории, полученные для сценариев а), б), в) при неполной информации о параметрах объекта

(сплошной линией обозначается движение целиs, пунктирной – перехватчика)

Анализ полученных данных позволяет сделать вывод, что управление, синтезированное с помощью интервальных методов глобальной условной оптимизации, для каждого из описанных сценариев успешно решило поставленную задачу перехвата при условии наличия неопределенности задания начальных состояний и параметров модели объекта управления, поскольку координаты цели и перехватчика с приемлемой точностью близки для различных начальных состояний из заданного множества $\boldsymbol{\Omega}$.

4.3. ЗАКЛЮЧЕНИЕ

Разработанные в первой главе интервальные методы оптимизации протестированы на ряде прикладных задач

- оптимизации технических систем: определение параметров сварной балки, сосуда высокого давления, редуктора и натяжной/компрессионной пружины.
- синтеза оптимального управления авиационно-космическими системами: задача преследования, управление солнечным парусом, стабилизация спутника, задача перехвата, задача групповой навигации, приземление гиперзвукового летательного аппарата.

Полученные результаты не противоречат уже известным решениям; а найденные приближенные решения прикладных задач полностью отвечают физической картине мира.

ЗАКЛЮЧЕНИЕ

Основным итогом диссертационной работы является разработка интервальных методов глобальной условной оптимизации и их применение для решения задач оптимизации технических систем и поиска оптимального управления (программного, с неполной обратной связью и управления по выходу) нелинейными детерминированными динамическими системами при неполной информации о состоянии и параметрах объекта, выразившаяся в следующих основных результатах:

1) разработаны интервальные алгоритмы оптимизации на основе инвертера (дихотомии целевого интервала, отсечки виртуальных значений, стохастической отсечки виртуальных значений, стохастических вырываний, обобщенный инверсный метод), предложена математическая модель интервальной ε-минимизации [27, 42, 63],
2) разработаны интервальные метаэвристические алгоритмы оптимизации (среднего пути, стохастической сетки, интервального разбросанного поиска, интервальный генетический алгоритм, интервальный метод взрывов, адаптивный интервальный алгоритм, самоорганизующийся интервальный алгоритм имитации эволюции колонии бактерий), доказаны теоремы о свойствах разработанных инверсных методов [43, 44, 48, 55, 56, 62, 64, 111],
3) разработаны интервальные алгоритмы поиска оптимального программного управления нелинейными непрерывными детерминированными системами [27, 42–44, 48, 55, 62–64, 111],
4) разработаны интервальные алгоритмы синтеза оптимального управления с неполной обратной связью и управления по выходу нелинейными непрерывными детерминированными системами [56],
5) разработан программный комплекс, реализующий интервальные методы глобальной условной оптимизации и алгоритмы их применения для поиска оптимального программного управления, оптимального в среднем управления пучками траекторий с неполной обратной связью и оптимального управления по выходу при неполной информации о состоянии и параметрах объекта с использованием математического моделирования поведения управляемых систем [52, 54],
6) разработанное алгоритмическое и программное обеспечение применено для решения задач оптимизации технических систем (определение оптимальных параметров сварной балки, сосуда высокого давления, редуктора, натяжной/компрессионной пружины) и систем управления авиационно-космической техникой (задачи о

преследовании, солнечном парусе, командной навигации, стабилизации спутника, перехвате, управлении гиперзвуковым летательным аппаратом) [27, 42–44, 48, 55, 56, 62–64, 111]

ПРИЛОЖЕНИЕ. ВВЕДЕНИЕ В ИНТЕРВАЛЬНЫЙ АНАЛИЗ

В приложении представлена более подробная информация о базовых терминах интервального анализа.

П.1. ОСНОВНЫЕ ПОНЯТИЯ ИНТЕРВАЛЬНОГО АНАЛИЗА

П.1.1. ИНТЕРВАЛЫ И ИНТЕРВАЛЬНЫЕ ВЕКТОРЫ

Основной идеей интервального анализа является окружение вещественных чисел интервалами, а вещественных векторов – интервальными векторами, или брусами. Условимся в дальнейшем для обозначения интервального вектора (бруса) использовать буквы латинского алфавита с полужирным начертанием: $\boldsymbol{x} = \boldsymbol{x}_1 \times \ldots \times \boldsymbol{x}_n = [\underline{\boldsymbol{x}}; \overline{\boldsymbol{x}}] = [\underline{\boldsymbol{x}_1}; \overline{\boldsymbol{x}_1}] \times \ldots \times [\underline{\boldsymbol{x}_n}; \overline{\boldsymbol{x}_n}] = \{x \in \mathbb{R}^n \mid \underline{\boldsymbol{x}} \le x \le \overline{\boldsymbol{x}}\}$. Множество брусов размерности n обозначим как $\mathbb{IR}^n$.

Для произвольного бруса $\boldsymbol{x}$ определены следующие характеристики:

- нижняя граница:

$$\min \boldsymbol{x} = \underline{\boldsymbol{x}} = (\underline{\boldsymbol{x}_1}, \ldots, \underline{\boldsymbol{x}_n})^{\mathrm{T}}, \tag{П.1}$$

- верхняя граница:

$$\max \boldsymbol{x} = \overline{\boldsymbol{x}} = (\overline{\boldsymbol{x}_1}, \ldots, \overline{\boldsymbol{x}_n})^{\mathrm{T}}, \tag{П.2}$$

- ширина:

$$\mathrm{wid}\, \boldsymbol{x} = \mathrm{wid}(\boldsymbol{x}) = \overline{\boldsymbol{x}} - \underline{\boldsymbol{x}} = (\overline{\boldsymbol{x}_1} - \underline{\boldsymbol{x}_1}, \ldots, \overline{\boldsymbol{x}_n} - \underline{\boldsymbol{x}_n}) \ge 0, \tag{П.3}$$

- радиус:

$$\mathrm{rad}\, \boldsymbol{x} = \mathrm{rad}(\boldsymbol{x}) = \frac{1}{2} \cdot (\overline{\boldsymbol{x}} - \underline{\boldsymbol{x}}) = \frac{1}{2} \cdot (\overline{\boldsymbol{x}_1} - \underline{\boldsymbol{x}_1}, \ldots, \overline{\boldsymbol{x}_n} - \underline{\boldsymbol{x}_n})^{\mathrm{T}}, \tag{П.4}$$

- средняя точка:

$$\mathrm{mid}\, \boldsymbol{x} = \mathrm{mid}(\boldsymbol{x}) = \frac{1}{2} \cdot (\overline{\boldsymbol{x}} + \underline{\boldsymbol{x}}) = \frac{1}{2} \cdot (\overline{\boldsymbol{x}_1} + \underline{\boldsymbol{x}_1}, \ldots, \overline{\boldsymbol{x}_n} + \underline{\boldsymbol{x}_n}). \tag{П.5}$$

Кроме этого, добавим следующую характеристику бруса (омега-характеристика):

$$\omega(\boldsymbol{x}) = \max_{i=1,\ldots,n} \mathrm{wid}(\boldsymbol{x}) = \max_{i=1,\ldots,n} (\overline{\boldsymbol{x}_1} - \underline{\boldsymbol{x}_1}, \ldots, \overline{\boldsymbol{x}_n} - \underline{\boldsymbol{x}_n}). \tag{П.6}$$

Следует упомянуть, что вырожденный интервальный вектор (с нулевым вектором ширины) соответствует обычному вещественному вектору, т.е. $\boldsymbol{x} = \mathrm{mid}(\boldsymbol{x}) = x \in \mathbb{R}^n$.

П.1.2. ИНТЕРВАЛЬНЫЕ АРИФМЕТИКИ

Среди большого количества различных интервальных арифметик остановимся подробнее на двух из них: на классической, т.к. она является необходимым базисом для построения более сложных арифметик, и на арифметики Кахана, которая использовалась в диссертационной работе.

Базовым объектом интервальной арифметики является интервал (интервальный вектор, или брус, из пространства $\mathbb{IR}^1$), т.е. объект, определяемый как $\boldsymbol{x} = [\underline{\boldsymbol{x}}; \overline{\boldsymbol{x}}] = \{x \in \mathbb{R} \mid \underline{\boldsymbol{x}} \le x \le \overline{\boldsymbol{x}}\}$.

В основе классической интервальной арифметики лежит принцип поэлементного определения операций, т.е.

$$\boldsymbol{x} \circ \boldsymbol{y} = \{x \circ y \mid x \in \boldsymbol{x}, y \in \boldsymbol{y}\}, \tag{П.7}$$

В соответствии с (П.7) можно определить следующие базовые арифметические операции:

- сложение:

$$\boldsymbol{x} + \boldsymbol{y} = [\underline{\boldsymbol{x}} + \underline{\boldsymbol{y}}; \overline{\boldsymbol{x}} + \overline{\boldsymbol{y}}], \tag{П.8}$$

- вычитание:

$$\boldsymbol{x} - \boldsymbol{y} = [\underline{\boldsymbol{x}} - \overline{\boldsymbol{y}}; \overline{\boldsymbol{x}} - \underline{\boldsymbol{y}}], \tag{П.9}$$

- умножение

$$\boldsymbol{x} \cdot \boldsymbol{y} = [\min\{\underline{\boldsymbol{x}} \cdot \underline{\boldsymbol{y}}, \underline{\boldsymbol{x}} \cdot \overline{\boldsymbol{y}}, \overline{\boldsymbol{x}} \cdot \underline{\boldsymbol{y}}, \overline{\boldsymbol{x}} \cdot \overline{\boldsymbol{y}}\}; \max\{\underline{\boldsymbol{x}} \cdot \underline{\boldsymbol{y}}, \underline{\boldsymbol{x}} \cdot \overline{\boldsymbol{y}}, \overline{\boldsymbol{x}} \cdot \underline{\boldsymbol{y}}, \overline{\boldsymbol{x}} \cdot \overline{\boldsymbol{y}}\}], \tag{П.10}$$

- деление

$$\boldsymbol{x} / \boldsymbol{y} = \boldsymbol{x} \cdot [1/\overline{\boldsymbol{y}}; 1/\underline{\boldsymbol{y}}], \ 0 \notin \boldsymbol{y}. \tag{П.11}$$

Таким образом, очевидно, что недостатком классической интервальной арифметики является отсутствие возможности деления на нульсодержащий интервал.

Интервальная арифметика Кахана позволяет этот недостаток устранить. В соответствии с ней операция деления может быть определена следующим образом:

$$\boldsymbol{x} / \boldsymbol{y} = \boldsymbol{x} \cdot \frac{1}{\boldsymbol{y}}, \text{ где}$$

$$\frac{1}{\boldsymbol{y}} = \begin{cases} \varnothing, \text{ если } \boldsymbol{y} = 0 = [0;0], \\ [1/\overline{\boldsymbol{y}}, 1/\underline{\boldsymbol{y}}], \text{ если } 0 \notin \boldsymbol{y}, \\ [1/\overline{\boldsymbol{y}}, +\infty], \text{ если } \underline{\boldsymbol{y}} = 0 \text{ и } \overline{\boldsymbol{y}} > 0, \\ [-\infty, 1/\underline{\boldsymbol{y}}], \text{ если } \underline{\boldsymbol{y}} < 0 \text{ и } \overline{\boldsymbol{y}} = 0, \\ [-\infty, +\infty], \text{ если } \underline{\boldsymbol{y}} < 0 \text{ и } \overline{\boldsymbol{y}} > 0. \end{cases} \tag{П.12}$$

Таким образом, интервальная арифметика Кахана расширяет множества интервалов, добавляя в него множества вида $]-\infty;a],[b;+\infty],]-\infty;+\infty[,\varnothing$, что в свою очередь позволяет оперировать неопределенностями более высокого уровня.

П.1.3. ИНТЕРВАЛЬНОЕ РАСШИРЕНИЕ ФУНКЦИЙ

Рассмотрим более подробно, как интерпретируется функция в терминологии интервального анализа. Для этого приведем несколько базовых понятий:

- интервальная функция $\boldsymbol{f}:\mathbb{IR}^n \to \mathbb{IR}$ называется *интервальный продолжением* вещественнозначной функции $f:\mathbb{R}^n \to \mathbb{R}$ на множестве $D \subset \mathbb{R}^n$, если $\forall x \in D: \boldsymbol{f}(x) = f(x)$,
- интервальная функция $\boldsymbol{f}:\mathbb{IR}^n \to \mathbb{IR}$ называется *интервальный расширением* вещественнозначной функции $f:\mathbb{R}^n \to \mathbb{R}$ на множестве $D \subset \mathbb{R}^n$, если является ее интервальным продолжением на этом множества, а также является монотонной по включению, т.е. $\forall \boldsymbol{x}, \boldsymbol{y} \subseteq D,\ \boldsymbol{x} \subseteq \boldsymbol{y} \Rightarrow \boldsymbol{f}(\boldsymbol{x}) \subseteq \boldsymbol{f}(\boldsymbol{y})$.

Таким образом, интервальное расширение позволяет получить оценку образа функции на некотором множестве. Однако приведенных определений недостаточно для того, чтобы алгоритмически определить способ расчета значения интервальной функции на произвольном брусе, так как пока определены не определены способы построения интервального расширения элементарных функций (модуля, степенной и показательной функции, логарифмической функции, а также тригонометрических и обратных тригонометрических функций). Интервальные расширения элементарных функций строятся на основе их поведения. Например, функция $\exp(x)$ является монотонной на всей области определения, что позволяет определить ее интервальное расширение следующим образом: $\mathbf{exp}(\boldsymbol{x}) = [\exp(\underline{\boldsymbol{x}}); \exp(\overline{\boldsymbol{x}})]$. Аналогично, путем анализа можно построить интервальные расширения для всех элементарных функций. Это, в свою очередь, позволит использовать *естественное интервальное расширение*, которое получается в результате замены всех аргументов на соответствующие интервальные величины, а арифметических операций и элементарных функций на их интервальные аналоги и расширения.

В дальнейшем, именно естественное интервальное расширение будет использоваться для формулировки всех постановок задач.

Следует отметить, что при использовании интервальных расширений имеет место так называемый эффект связанности (зависимости), который является причиной огрубления интервальных оценок, полученных с помощью расширения. Для его демонстрации

рассмотрим следующую вещественнозначную функцию $f(x) = x - x$ и построим ее интервальное расширение. Очевидно, что $\forall x: f(x) = x - x = 0$. Тем не менее для ее интервального расширения при использовании выбранной ранее арифметики Кахана для невырожденных интервалов (т.е. интервалов ненулевой ширины) мы будем получать отличные от нуля значения. Действительно: $\boldsymbol{f}([1;2]) = [1;2] - [1;2] = [-1;1] \neq 0$. Стоит также отметить, что при уменьшении ширины бруса данный эффект будет слабеть.

П.2. ИНВЕРТЕР

Инвертером [97] называется функция $INV(f, \boldsymbol{s}, \boldsymbol{l}, \varepsilon)$, которая по заданному интервалу $\boldsymbol{l}$, функции f и параметру точности ε находит в исходной области поиска $\boldsymbol{s}$ множество брусов $\mathcal{P}$ такое, что $\forall \boldsymbol{x} \in \mathcal{P}$:

$$\boldsymbol{x} \subseteq \boldsymbol{s}, \omega(\boldsymbol{x}) \geq \varepsilon, \boldsymbol{f}(\boldsymbol{x}) \subseteq \boldsymbol{l} \text{ или } \boldsymbol{x} \subseteq \boldsymbol{s}, \omega(\boldsymbol{x}) < \varepsilon, \boldsymbol{f}(\boldsymbol{x}) \cap \boldsymbol{l} \neq \varnothing. \quad (\text{П.13})$$

Инвертер используется как базовая составляющая методов оптимизации для отсева брусов, которые не содержат точек глобального экстремума.

Алгоритм SIVIA

Данный алгоритм разработан Жоленом и Вальтером [97] в 1993. С его помощью можно приближенно решить задачу обращения (инвертирования) множества, которая заключается в следующем: есть некоторая функция $f: \mathbb{R}^n \to \mathbb{R}$, некоторый интервал $\boldsymbol{l}$, необходимо найти множество $\mathbb{X} \subset \mathbb{R}^n$, содержащее все точки $x \in \mathbb{R}^n$, для которых значение функции $f(x)$ принадлежит интервалу $\boldsymbol{l}$:

$$\mathbb{X} = \{x \in \mathbb{R}^n \mid f(x) \in \boldsymbol{l}\}. \quad (\text{П.14})$$

Алгоритм SIVIA позволяет найти множество $\mathcal{P}$ интервальных векторов, таких что

$$\mathbb{X} \subseteq \bigcup_{i=1}^{|\mathcal{P}|} \boldsymbol{p}^i, \quad (\text{П.15})$$

где $|\cdot|$ - мощность множества. Для работы алгоритма необходима некоторая начальная (возможно очень большая) область поиска $\boldsymbol{s}$ (предполагается, что $\mathbb{X} \subseteq \boldsymbol{s}$) и параметр точности $\varepsilon > 0$.

Опишем алгоритм SIVIA.

Шаг 1. Задать функцию f, область поиска (брус) $\boldsymbol{s} \in \mathbb{IR}^n$, интервал $\boldsymbol{l}$ и параметр точности ε. Создать пустое множество брусов $\mathcal{P}$ и множество $\mathcal{X} = \{\boldsymbol{s}\}$.

Шаг 2. Для каждого бруса $\boldsymbol{x} \in \mathcal{X}$ найти значение естественного интервального расширения $\boldsymbol{f}(\boldsymbol{x})$. Проверить выполнение следующих условий (сам брус $\boldsymbol{x}$ после проверки условий удаляется из $\mathcal{X}$):

- если $\boldsymbol{f}(\boldsymbol{x}) \cap \boldsymbol{l} = \varnothing$, то брус $\boldsymbol{x}$ не добавляется в множество $\mathcal{P}$ (случай «б» на рис. П.1);
- если $\boldsymbol{f}(\boldsymbol{x}) \subseteq \boldsymbol{l}$ или $\boldsymbol{f}(\boldsymbol{x}) \cap \boldsymbol{l} \neq \varnothing, \omega(\boldsymbol{x}) < \varepsilon$, то добавить брус $\boldsymbol{x}$ в множество $\mathcal{P}$ (случаи «в» и «г» на рис. П.1);
- в противном случае задать интервальные векторы $\boldsymbol{x}^1 = \boldsymbol{x}_1 \times \ldots \times [\underline{\boldsymbol{x}_k}; \mathrm{mid}(\boldsymbol{x}_k)] \times \ldots \times \boldsymbol{x}_n$ и $\boldsymbol{x}^2 = \boldsymbol{x}_1 \times \ldots \times [\mathrm{mid}(\boldsymbol{x}_k); \overline{\boldsymbol{x}_k}] \times \ldots \times \boldsymbol{x}_n$, где k – наименьший номер компоненты бруса $\boldsymbol{x}$ с наибольшей шириной и добавить их в множество $\mathcal{X}$ (процедура бисекции; случай «а» на рис. П.1).

Шаг 3. Если $\mathcal{X} = \varnothing$, то перейти к шагу 3, в противном случае – к шагу 4.

Шаг 4. Вернуть $\mathcal{P}$.

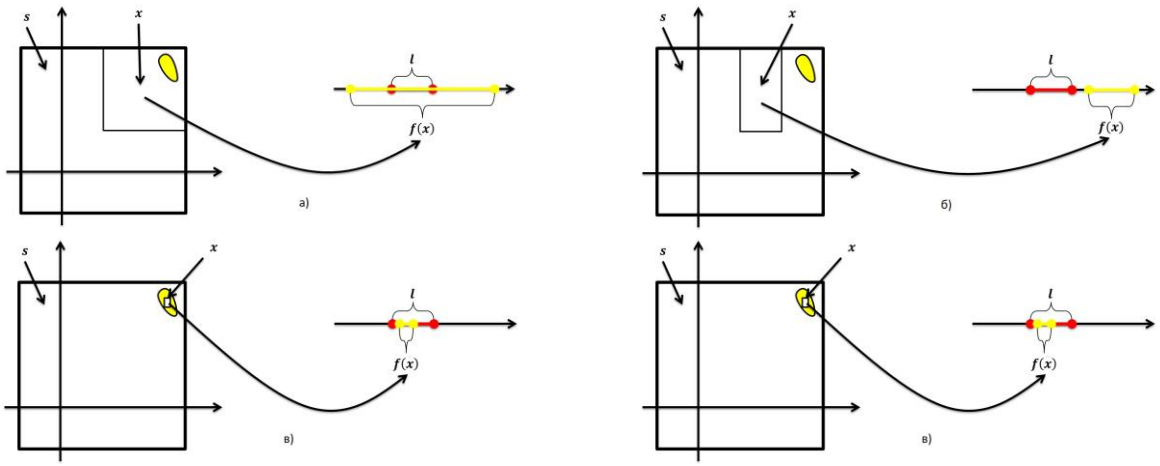


Рис. П.1. Иллюстрация случаев алгоритма SIVIA
(желтым выделено множество $\mathbb{X} = \{x \mid f(x) \in \boldsymbol{l}\}$)

На рис. П.2 видно, что бисекции подвергается лишь тот брус, в котором точно содержится хотя бы часть множества $\mathbb{X}$ (либо тот интервальный вектор, который удовлетворяет условиям точности). В данном случае $n = 2$, а бисекция реализуется по компоненте с наибольшей шириной (если ширина компонент одинакова, то берется компонента с наименьшим номером).

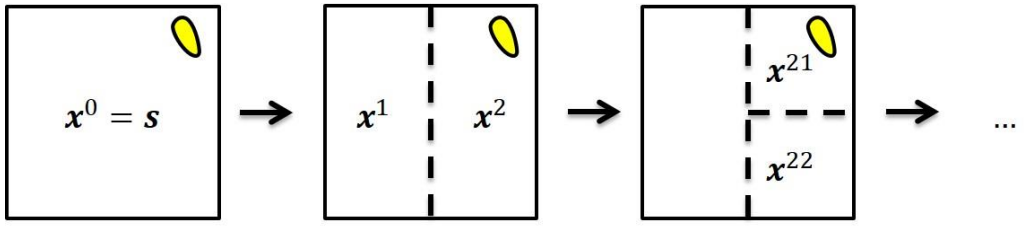


Рис. П.2. Схема генерации брусов

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Алефельд Г., Херцбергер Ю. Введение в интервальные вычисления / Г. Алефельд, Ю. Херцбергер, Москва: Мир, 1987.

2. Брадис В.М. Теория и практика вычислений. Пособие для высших педагогических учебных заведений / В.М. Брадис, Москва: Учпедгиз, 1937.

3. Гусев Ю.М. [и др.]. Анализ и синтез линейных интервальных динамических систем (состояние проблемы). Часть I // Изв. РАН. Техн. кибернетика. 1991. № 1. С. 3–23.

4. Гусев Ю.М. [и др.]. Анализ и синтез линейных интервальных динамических систем (состояние проблемы). Часть II // Изв. РАН. Техн. кибернетика. 1991. № 2. С. 3–30.

5. Добронец Б.С. Интервальная математика / Б.С. Добронец, Красноярск:, 2007.

6. Ефанов В.Н., Крымский В.К., Тляшов Р.З. Алгоритмическая процедура синтеза многосвязных систем с интервальными характеристическими полиномами // 1989. 12 с.

7. Захаров А.В., Шокин Ю.И. Синтез систем управления при интервальной неопределенности параметров их математических моделей // ДАН СССР. 1988. № 2 (299). С. 292–295.

8. Ивлев Р.С., Соколова С.П. Построение векторного управления многомерным интервально заданным объектом // Вычислительные технологии. 1999. № 4 (4). С. 3–13.

9. Калмыков С.А., Шокин Ю.И., Юлдашев З.Х. Методы интервального анализа / С.А. Калмыков, Ю.И. Шокин, З.Х. Юлдашев, Новосибирск: Наука, 1986.

10. Канторович Л.В. О некоторых новых подходах к вычислительным методам и обработке наблюдений // Сибирский Математический Журнал. 1962. № 5 (3). С. 701–709.

11. Крылов И.А. Численное решение задачи об оптимальной стабилизации спутника // Вычислительная математика и физика. 1986. № 8(1). С. 203–208.

12. Меньшиков Г.Г. Интервальный анализ и методы вычислений: Конспект лекций. Выпуск 3. Интервализация приближенных формул. Численное суммирование рядов / Г.Г. Меньшиков, Санкт-Петербург:, 1996.

13. Меньшиков Г.Г. Интервальный анализ и методы вычислений: Конспект лекций. Выпуск 1. Введение в интервальную организацию вычислений / Г.Г. Меньшиков, Санкт-Петербург:, 1996.

14. Меньшиков Г.Г. Интервальный анализ и методы вычислений: Конспект лекций. Выпуск 6. Локализующее вычисление интегралов / Г.Г. Меньшиков, Санкт-Петербург:, 1998.

15. Меньшиков Г.Г. Интервальный анализ и методы вычислений: Конспект лекций. Выпуск 9. Элементы локализующего интегрирования дифференциальных уравнений / Г.Г.

Меньшиков, Санкт-Петербург:, 1998.
16. Морозов М.В. Условия робастной устойчивости линейных нестационарных систем управления с интервальными ограничениями // Проблемы управления. 2009. № 3. С. 23–26.
17. Панов Н.В. Разработка рандомизированных алгоритмов в интервальной глобальной оптимизации // Диссертация канд. ф.-м. наук. Конструкторско-технологический институт вычислительной техники СО РАН, Новосибирск, 2012. С. 178.
18. Панов Н.В., Шарый С.П. Интервальный эволюционный алгоритм для поиска глобального оптимума // Известия Алтайского государственного университета. 2011. № 1(69) (2). С. 108–113.
19. Пановский В.Н. Формирование модифицированной интервальной арифметики и ее реализация в виде комплекса программ интервального анализа (Panovskiy V.N. Formation of modified interval arithmetic and its implementation as a complex of programs of interval analysis) // Актуальные проблемы авиационных и аэрокосмических систем (Actual problems of aviation and aerospace systems). 2010. С. 154–155, 165–166.
20. Пановский В.Н. Применение интервальной арифметики для решения систем линейных алгебраических уравнений // Межвузовский сборник научных трудов «Теоретические вопросы вычислительной техники и программного обеспечения». 2010. С. 137–142.
21. Пановский В.Н. Формирование модифицированной интервальной арифметики и ее реализация в виде комплекса программ интервального анализа // II Международная научно-практическая конференция «Научно-техническое творчество молодежи - путь к обществу, основанному на знаниях»: Сборник научных докладов. 2010. С. 435–436.
22. Пановский В.Н. Комплекс программ для интервального анализа на языке C# // Труды VII Всероссийской конференции студентов, аспирантов и молодых ученых «Технологии Microsoft в теории и практике программирования». 2010. С. 131–132.
23. Пановский В.Н. Прикладное применение интервальных алгоритмов sivia и imagesp // 9-я Международная конференция «Авиация и космонавтика - 2010». 2010. С. 323–324.
24. Пановский В.Н. Применение интервальных алгоритмов инвертирования множества значений (sivia) и оценивания образа множества (image_sp) // Межвузовский сборник научных трудов «Теоретические вопросы вычислительной техники и программного обеспечения»2. 2011. (1). С. 27–32.
25. Пановский В.Н. Реализация сжимающих операторов в виде программы на языке C# // Научно-практическая конференция молодых ученых и студентов МАИ «Инновации в авиации и космонавтике - 2011». 2011. С. 109.
26. Пановский В.Н. Реализация методов глобальной оптимизации, использующих аппарат

интервального анализа // 10-я Международная конференция «Авиация и космонавтика - 2011». 2011. С. 239–240.
27. Пановский В.Н. Применение аппарата интервального анализа для поиска глобального экстремума функций // Труды МАИ. 2012. № 51.
28. Пановский В.Н. Реализация методов глобальной оптимизации, использующих аппарат интервального анализа (Realization of methods of global optimization using interval analysis) // Актуальные проблемы авиационных и аэрокосмических систем (Actual problems of aviation and aerospace systems). 2012. № 155–156, 166–167.
29. Пановский В.Н. Сравнительный анализ интервальных методов глобальной условной оптимизации // VI Межвузовский сборник научных трудов «Прикладная информатика и математическое моделирование». 2012. С. 73–83.
30. Пановский В.Н. Сравнительный анализ интервальных методов глобальной условной оптимизации // Московская молодежная научно-практическая конференция «Инновации в авиации и космонавтике – 2012». Сборник тезисов. 2012. С. 246–247.
31. Пановский В.Н. Применение интервального анализа для решения задачи о максимизации стоимости предприятия при ограниченных параметрах // Научный альманах. Выпуск 16: Материалы VIII научно-практической конференции молодых ученых и студентов «Инновационный менеджмент в аэрокосмической промышленности». 2012. С. 183–189.
32. Пановский В.Н. Метод стохастической отсечки мнимых значений для поиска глобального экстремума // Материалы Юбилейной 50-й Международной научной студенческой конференции «Студент и научно-технический прогресс»: Математика. 2012. С. 235.
33. Пановский В.Н. Интервальные алгоритмы поиска глобального экстремума функций // Труды Международной научно-методической конференции «Информатизация инженерного образования» - ИНФОРИНО-2012. 2012. С. 224–227.
34. Пановский В.Н. Интервальные алгоритмы нахождения оптимального управления с полной обратной связью детерминированными непрерывными системами // Тезисы докладов 11-й Международной конференции «Авиация и космонавтика – 2012». 2012. С. 391–392.
35. Пановский В.Н. Интервальные алгоритмы нахождения оптимального управления с полной обратной связью детерминированными дискретными системами // Сборник тезисов докладов Всероссийской молодежной научно-технической конференции «Прикладные научно-технические проблемы современной теории управления системами и процессами». 2012. С. 48.
36. Пановский В.Н. Применение интервальных методов глобальной условной оптимизации в

экономических задачах // Научный альманах. Выпуск 17: Материалы IX научно-практической конференции молодых ученых и студентов «Инновации в экономике и менеджменте аэрокосмической промышленности». 2013. С. 205–213.

37. Пановский В.Н. Синтез оптимального программного управления нелинейным динамическим объектом // Сборник тезисов докладов московской молодежной научно-практической конференции «Инновации в авиации и космонавтике – 2013». 2013. С. 293.

38. Пановский В.Н. Эвристические интервальные методы глобальной условной оптимизации // Материалы 51-й Международной научной студенческой конференции «Студент и научно-технический прогресс»: Математика. 2013. С. 177.

39. Пановский В.Н. Интервальный метод глобальной условной оптимизации. Метод стохастических вырываний // Материалы XVIII Международной конференции по вычислительной механике и современным прикладным и программным системам. 2013. С. 850–851.

40. Пановский В.Н. Прикладное применение интервального метода взрывов // Тезисы докладов 12-й Международной конференции «Авиация и космонавтика – 2013». 2013. С. 613–615.

41. Пановский В.Н. Интервальный генетический алгоритм глобальной условной оптимизации // Сборник докладов Четвертой научно-технической конференции молодых ученых и специалистов «Научные чтения к 105-летию со дня рождения академика А.А. Расплетина». 2013. С. 601–607.

42. Пановский В.Н. Обобщенный инверсный интервальный метод глобальной условной оптимизации // Научный вестник МГТУ ГА. 2014. № 207. С. 17–24.

43. Пановский В.Н. Интервальный генетический алгоритм глобальной условной оптимизации // Успехи современной радиоэлектроники. 2014. № 4. С. 71–75.

44. Пановский В.Н. Прикладное применение интервального метода взрывов // Труды МАИ. 2014. № 73.

45. Пановский В.Н. Разработка интервальных методов оптимизации нелинейных детерминированных динамических систем // Сборник тезисов докладов московской молодежной научно-практической конференции «Инновации в авиации и космонавтике – 2014». 2014. С. 213–214.

46. Пановский В.Н. Применение интервального метода взрывов для оптимизации стоимостных показателей предприятия // Научный альманах. Выпуск 19: Материалы X научно-практической конференции молодых ученых и студентов «Инновации в экономике и менеджменте аэрокосмической промышленности». 2014. С. 194–202.

47. Пановский В.Н. Прикладное применение интервального метода взрывов для решения задачи командной навигации // Актуальные вопросы развития систем и средство воздушно-космической обороны. Сборник докладов Пятой научно-технической конференции молодых ученых и специалистов. 2014. С. 650–656.
48. Пановский В.Н. Прикладное применение интервального метода взрывов для решения задачи командной навигации // Успехи современной радиоэлектроники. 2015. № 3. С. 207–212.
49. Пановский В.Н. Применение интервального метода взрывов для прогнозирования временных рядов // Материалы XI научно-практической конференции молодых ученых и студентов «Инновации в экономике и менеджменте в авиации». 2015. С. 189–195.
50. Пановский В.Н. Сравнительная характеристика интервальных методов оптимизации нелинейных детерминированных динамических моделей // Аннотированный сборник материалов Всероссийской научно-технической конференции «Расплетинские чтения – 2015». 2015. С. 119.
51. Пановский В.Н. Прикладное применение эвристических интервальных алгоритмов оптимизации для решения задач перехвата и групповой навигации // Тезисы докладов II Международной конференции «Инжиниринг & Телекоммуникации – EN&T 2015». 2015. С. 119–121.
52. Пановский В.Н. Программа поиска оптимального управления летательными аппаратами с использованием интервальных методов глобальной оптимизации // Федеральная служба по интеллект. собственности. Св-во о гос. регистрации программы для ЭВМ № 2015661635. 2015.
53. Пановский В.Н. Прикладное применение адаптивного интервального алгоритма для управления нелинейным динамическим объектом // Аннотированный сборник материалов Всероссийской научно-технической конференции «Расплетинские чтения – 2016». 2016. С. 120.
54. Пановский В.Н. Программа синтеза оптимального в среднем управления летательными аппаратами с неполной обратной связью с использованием интервальных методов глобальной оптимизации // Федеральная служба по интеллект. собственности. Св-во о гос. регистрации программы для ЭВМ № 2016610641. 2016.
55. Пановский В.Н., Пантелеев А.В. Прикладное применение интервальных методов оптимизации для решения задачи групповой навигации // Успехи современной радиоэлектроники. 2016. № 2. С. 177–182.
56. Пановский В.Н., Пантелеев А.В. Метаэвристические интервальные методы поиска

оптимального в среднем управления нелинейными детерминированными системами при неполной информации о ее параметрах // Известия РАН. Теория и системы управления. 2017. № 1. С. 44–55.

57. Пантелеев А.В. Теория управления в примерах и задачах / А.В. Пантелеев, Москва: Высшая школа, 2003.

58. Пантелеев А.В., Пановский В.Н. Применение интервальной арифметики для решения систем линейных алгебраических уравнений // XLVI Всероссийская конференция по проблемам математики, информатики, физики и химии: Тезисы докладов секции математики и информатики. 2010. С. 29–30.

59. Пантелеев А.В., Пановский В.Н. Реализация сжимающих операторов в виде программы на языке C# (Panovskiy V.N., Panteleyev A.V. Implementation of contractors in a form of a program in C#) // Актуальные проблемы авиационных и аэрокосмических систем (Actual problems of aviation and aerospace systems). 2011. С. 154, 170–171.

60. Пантелеев А.В., Пановский В.Н. Комплекс программных средств интервального анализа: реализация алгоритмов нахождения образа и прообраза множеств // Сборник научных трудов «Проблемы авиастроения, космонавтики и ракетостроения». 2012. С. 303–309.

61. Пантелеев А.В., Пановский В.Н. Интервальные методы поиска глобального условного экстремума // Вопросы создания аэрокосмических и ракетных летательных аппаратов. 2013. С. 188–193.

62. Пантелеев А.В., Пановский В.Н. Разработка интервальных метаэвристических методов минимизации для поиска оптимального программного управления // Управление большими системами. 2016. № 60. С. 41–62.

63. Пантелеев А.В., Пановский В.Н. Применение обобщенного инверсного интервального метода глобальной условной оптимизации в задаче поиска оптимального программного управления // Вестник МГТУ им. Н.Э. Баумана. Серия «Приборостроение». 2016. № 1(106). С. 33–50.

64. Пантелеев А.В., Пановский В.Н., Короткова Т.И. Интервальный метод взрывов и его применение к задаче моделирования и оптимизации движения гиперзвукового летательного аппарата // Вестник Южно-Уральского государственного университета, серия «Математическое моделирование и программирование». 2016. № 3 (9). С. 55–67.

65. Пантелеев А.В., Рыбаков К.А. Прикладной вероятностный анализ нелинейных систем управления спектральным методом / А.В. Пантелеев, К.А. Рыбаков, Москва: МАИ-ПРИНТ, 2010.

66. Пантелеев А.В., Рыбаков К.А. Анализ нелинейных стохастических систем управления в

классе обобщенных характеристических функций // Автоматика и телемеханика. 2011. № 11. С. 183–194.

67. Самойленко В.И. Техническая кибернетика / В.И. Самойленко, Москва: МАИ, 1994.

68. Смагина Е.М., Дугарова И.В. Синтез модального регулятора для систем с неопределенными параметрами // 1987. 37 с.

69. Смагина Е.М., Моисеев А.Н. Слежение за полиномиальным сигналом в интервальной динамической системе // Вычислительные технологии. 1998. № 1 (3). С. 67–74.

70. Харитонов В.Л. Об асимптотической устойчивости положения равновесия семейства систем линейных дифференциальных уравнений // Дифференциальные уравнения. 1978. № 2(11) (14). С. 2086–2088.

71. Хлебалин Н.А. Аналитический синтез регуляторов в условиях неопределенности параметров объекта // Аналитические методы синтеза регуляторов: Межвуз. научн. сб. 1981. С. 107–123.

72. Хлебалин Н.А. Синтез интервальных регуляторов в задаче модального управления // Аналитические методы синтеза регуляторов. Саратов: Саратовский политехнический ин-т. 1988. С. 26–30.

73. Шарый С.П. О разрешимости линейной задачи о допусках // Interval Computations. 1991. № 1. С. 92–97.

74. Шарый С.П. Новый подход в интервальной глобальной оптимизации // Труды XII Байкальской международной конференции «Методы оптимизации и их приложения». 2001. (1). С. 289–295.

75. Шарый С.П. Рандомизированные алгоритмы в интервальной глобальной оптимизации // Сибирский журнал вычислительной математики. 2008. № 4 (11). С. 457–474.

76. Шарый С.П. Конечномерный интервальный анализ / С.П. Шарый, Новосибирск: XYZ, 2010.

77. Шарый С.П. Курс вычислительных методов / С.П. Шарый, Новосибирск: Новосибирский гос. ун-т, 2010.

78. Шашихин В.Н. Методы интервального анализа в синтезе робастного управления // Вычислительные технологии. 2001. № 6 (6). С. 118–123.

79. Шокин Ю.И. Интервальный анализ / Ю.И. Шокин, Москва: Наука, 1981.

80. Alefeld G., Lohner R. On Higher Order Centered Forms // Computing. 1985. № 35. С. 177–184.

81. Alefeld G., Mayer G. The Cholesky method for interval data // Linear Algebra and Its Applications. 1993. № C (194). С. 161–182.

82. Bäck T., Fogel D.B., Michalewicz Z. Evolutionary Computation 1. Basic Algorithms and Operators / T. Bäck, D.B. Fogel, Z. Michalewicz, Bristol and Priladelphia: Institute of Physics Publishing, 2000.
83. Behnke H., Goersich F. Inclusions for Eigenvalues of Selfadjoint Problems // Topics in Validated Computations. 1994. С. 277–322.
84. Cagnina L.C., Esquivel S.C. Solving Engineering Optimization Problems with the Simple Constrained Particle Swarm Optimizer // Informatica. 2008. № 32. С. 319–326.
85. Cornelius H., Lohner R. Computing the Range of Values of Real Functions with Accuracy Higher than Second Order // Computing. 1984. № 33. С. 331–347.
86. Dussel R. Einschließung des Minimalpunktes einer streng konvexen Funktion auf einem n-dimensiomalen Quader 1972.
87. Dwyer P.S. Linear Computations / P.S. Dwyer, New York: John Wiley & Sons, 1951.
88. Eijgenraam P. The Solution of Initial Value Problems Using Interval Arithmetic. Formulation and Analysis of an Algorithm 1981.
89. Floudas C.A., Pardalos P.M. Encyclopedia of Optimization / C.A. Floudas, P.M. Pardalos, New York: Springer, 2009.
90. Gardenes T., Trepat A. Fundamentals of SIGLA, an Interval Computing System Over the Completed Set of Intervals // Computing. 1980. № 24. С. 161–179.
91. Golinski J. An Adaptive Optimization System Applied to Machine Synthesis // Mech. Mach. Theory. 1973. № 8(3). С. 419–436.
92. Hansen E. Interval Arithmetic in Matrix Computations Part I // SIAM Journal on Numerical Analysis. 1965. № 2 (2). С. 308–320.
93. Hansen E., Walster G.W. Global Optimization Using Interval Analysis / E. Hansen, G.W. Walster, New York: Marcel Dekker, 2004.
94. Hu X., Eberhart R.C., Shi Y. Engineering Optimization with Particle Swarm // Proceedings of the IEEE Swwarm Intelligence Symposium. 2003. С. 53–57.
95. Hughes G.W., McDonald M., McInnes C.R. Terrestrial Planet Sample Return Using Solar Sail Propulsion // Acta Astronautica. 2006. № 59(8-11). С. 797–806.
96. Jansson C. Rigorous Sensitivity Analysis for Real Symmetric Matrices with Interval Data // Computer Arithmetic, Scientific Computation and Mathematical Modeling, IMACS Annals on Computing and Applied Mathematics. 1994. С. 293–316.
97. Jaulin L. [и др.]. Applied Interval Analysis / L. Jaulin, M. Kieffer, O. Didrit, E. Walter, London: Springer-Velag, 2001.
98. Jeon I.-S., Lee J.-I. Homing Guidance Law for Cooperative Attack of Multiple Missiles //

Journal of Guidance, Control and Dynamics. 2010. № 1 (33). С. 275–280.
99. Kearffot R.B. Rigorous Global Search: Continuos Problems / R.B. Kearffot, Dordrecht: Kluwer, 1996.
100. Krawczyk R. Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken // Computing. 1969. № 4. С. 187–201.
101. Kreinovich V., Kearffot R.B. Beyond Convex? Global Optimization is Feasible Only for Convex Objective Functions: a Theorem // Journal of Global Optimization. 2005. № 4 (33). С. 617–624.
102. Lohner R. Enclosing All Eigenvalues of Symmetric Matrices // Accurate Numerical Algorithms. 1989. № 1. С. 87–103.
103. McInnes C.R., Hughes G.W. Low Cost Mercury Orbiter and Sample Return Missions Using Solar Sail Propulsion // Aeronautical Journal. 2003. № 107(1074). С. 469–478.
104. Miranda C. Un’ osservazion su un teorema di Brouwer // Bol. Un. Mat. Ital. Ser. 1941. № 3 (2). С. 5–7.
105. Moore R.E. Interval Analysis / R.E. Moore, Englewood Cliffs: Prentice Hall, 1966.
106. Moore R.E. Methods and Applications of Interval Analysis / R.E. Moore, Philadelphia: SIAM, 1979.
107. Nakao M. State of the Art for Numerical Computations with Guaranteed Accuracy // Math. 1998. № 48. С. 323–338.
108. Nedialkov N.S. Computing Rigorous Bounds on the Solution of an Initial Value Problem for an Ordinary Differential Equation 1999.
109. Nickel K. Über die Notwendigkeit einer Fehlerschranken-Arithmetic für Rechenautomaten // Numerische Mathematik. 1966. С. 69–79.
110. Panovskiy V. Interval Methods for Global Unconstrained Optimization: a Software Package // Book of Abstracts of 15th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetics and Verified Numerics. 2012. С. 131–132.
111. Panteleev A., Panovskiy V. Self-Organizing Interval Bacterial Colony Evolution Optimization Algorithm // Advances in Intelligent Systems and Computing. 2016. (450). С. 451–463.
112. Plum M. Inclusion Methods for Elliptic Boundary Value Problems // Topics in Validated Computations. 1994. С. 323–379.
113. Ragsdell K., Phillips D. Optimal Design of a Class of Welded Structures using Geometric Programming // Journal of Engineering for Industry. 1976. № 98(3). С. 1021–1025.
114. Ratschek H., Rokne J. New Computer Methods for Global Optimization / H. Ratschek, J. Rokne, Chichester: Horwood, 1988.

115. Rump S. Solving Algebraic Problems with High Accuracy // A New Approach to Scientific Computation. 1983. С. 323–379.
116. Sandgren E. Nonlinear Integer and Discrete Programming in Mechnical Design Optimization // Journal of Mechanical Design. 1990. № 2 (112). С. 223–229.
117. Shary S.P. Randomized Algorithms in Interval Global Optimization // Numerical Analysis and Applications. 2008. № 4 (1). С. 376–389.
118. Singh B., Bhattachary R. Optimal Guidance of Hypersonic Vehicles Using B-Splines and Galerkin Projection // Guidance, Navigation and Control .Conference and Exhibit. 2008. С. 5749–5759.
119. Smagina Y., Brewer I. Using interval arithmetic for robust state feedback design // Systems & Control Letters. 2008. № 4 (1). С. 376–389.
120. Sunaga T. Theory of an Interval Algebra and its Application to Numerical Analysis // RAAG Memoirs. 1958. С. 547–564.
121. Tewari A. Advanced Control of Aircraft, Spacecraft and Rockets / A. Tewari, New York: A John Wiley & Sons, 2011.
122. Wang Y., Zhu M., Wei Y. Solar Sail Spacecraft Trajectory Optimization Based on Improved Imperialist Competitive Algorithm // Proceedings of the 10th World Congress on Intelligent Control and Automation. 2012. С. 191–195.
123. Warmus M. Calculus of Approximations // Bulletin de l'Academie Polonaise de Sciences. 1956. № 5 (4). С. 253–257.
124. Young R.C. Algebra of Many-Valued Quantities // Mathematische Annalen. 1931. (104). С. 260–290.
125. Neumaier A. Interval Methods for Systems and Equations / A. Neumaier, Cambridge: Cambridge University Press, 1990.