МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет Механико-математический факультет

Кафедра Математического моделирования

Направление подготовки Механика и математическое моделирование

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА

Андросов Артем Станиславович

Тема работы: Создание библиотеки интервальных вычислений для языка программирования Python

«К защите допущен»
Заведующий кафедрой
д. ф.-м. н., проф.

Барахнин В.Б. / _____
(Фамилия И.О.) (Подпись)

« . . » . . . . . 2022 г.

Научный руководитель
д-р физ.-мат.наук, проф.
в. н. с. ФИЦ ИВТ

Шарый С.П. / _____
(Фамилия И.О.) (Подпись)

« . . » . . . . . 2022 г.

Дата защиты: « . . » . . . . . 2022 г.

Новосибирск, 2022 г.

Оглавление

Введение

Работа посвящена созданию библиотеки программ на языке Python, реализующей численные методы интервального анализа для решения ряда стандартных математических задач. Это новейшие методы для определения существования решений как квадратных, так и переопределённых интервальных систем линейных алгебраических уравнений, вычисление интервалов, в которых лежат искомые параметры и визуализации множеств решений интервальных уравнений и систем уравнений. Кроме того, в библиотеке имеются также реализации известных интервальных методов для решения нелинейных систем.

Постановка задачи вкратце такова: необходимо реализовать инструмент, основанный на мощном математическом аппарате, для манипуляций с интервальными линейными системами в условиях вещественности решения.

Мотивацией для данной работы послужил тот факт, что в языке Python отсутствуют хорошо развитые и известные интервальные библиотеки в которых реализована подобная функциональность. Последняя созданная библиотека, в которой реализованы интервальные методы, уже не поддерживается её разработчиками. Кроме того, среди огромного количества программных модулей, реализованных к настоящему моменту для Python, нет хороших продуктов для визуализации множеств решений линейных неравенств $A\,x \leq b$ для случаев двух и трёх переменных.

Далее для простоты мы будем использовать общепринятое слэнговое название языка, т. е. вместо Python будем попросту говорить «Питон».

Один из итогов работы — появление возможности работы с векторами, которая была реализована внутри интервального класса. Это значительно упрощает работу с кодом, а также ускоряет вычисления, что особенно полезно для пользователей данной библиотеки.

Кратко охарактеризуем математическую «начинку» библиотеки. Для исследования разрешимости интервальных систем уравнений были реализованы распознающие функционалы Tol и Uni. Для оптимизации функционала Tol был переписан код известной программы `tolsolvty`, а для Uni используется метод Нелдера-Мида. В случае, когда система

нелинейна, обобщён метод для распознавания сильной согласованности. Однако стоит отметить, что в общем случае функционал Tol не обязан являться вогнутым (или, в более слабой постановке, квазивогнутым), и поэтому полученный результат будет сильно зависеть от начального приближения.

Для решения различных задач, возникающих в связи с интервальными системами линейных алгебраических уравнений (ИСЛАУ) реализованы несколько методов, включая те, которые способны решать переопределённые системы. В частности, реализованы метод Рона и методы дробления решений (PSS-методы). Если пользователь данного модуля хочет получить оптимальную оценку, то ему следует воспользоваться гибридным методом дробления решений с делением из арифметики Кахана, т. е. решение будет получено даже в том случае, когда алгоритм встретится с нульсодержащим делителем.

Для визуализации множеств решений интервальных линейных уравнений и линейных неравенств можно воспользоваться функциями в которых заложен метод граничных интервалов. Они способны отрисовывать, как двумерные, так и трёхмерные множества решений.

Глава 1

Интервальный анализ

1.1. Определения и обозначения

Интервалом $[a, b]$ вещественной оси $\mathbb{R}$ будем называть множество всех чисел, расположенных между числами a и b, включая их самих, т.е.

$$[a, b] := \{x \in \mathbb{R} \,|\, a \leq x \leq b\}.$$

Система обозначений в работе соответствует неформальному международному стандарту в интервальном анализе [5]. В частности, интервалы и интервальные величины обозначаются жирным шрифтом, к примеру, $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, ..., в то время, как точечные значения никаким специальным образом не выделяются. Кроме того, будем обозначать концы интервала (т.е. его правую и левую границы), как $\underline{\boldsymbol{a}}$ и $\overline{\boldsymbol{a}}$, т.е. $\boldsymbol{a} = [\underline{\boldsymbol{a}}, \overline{\boldsymbol{a}}]$.

Прежде всего, определим наиболее важные и часто встречающиеся арифметические операции между интервалами.

$\boldsymbol{a} * \boldsymbol{b} = \{a * b \mid a \in \boldsymbol{a}, b \in \boldsymbol{b}\}$, где $* \in \{+, -, \cdot, /\}$, — основной принцип определения классической интервальной арифметики.

Предложение 1.1.1 *Конструктивные определения интервальных арифметических операций имеют вид:*

$$\boldsymbol{a} + \boldsymbol{b} = [\underline{\boldsymbol{a}} + \underline{\boldsymbol{b}},\ \overline{\boldsymbol{a}} + \overline{\boldsymbol{b}}], \tag{1.1}$$

$$\boldsymbol{a} - \boldsymbol{b} = [\underline{\boldsymbol{a}} - \overline{\boldsymbol{b}},\ \overline{\boldsymbol{a}} - \underline{\boldsymbol{b}}], \tag{1.2}$$

$$\boldsymbol{a} \cdot \boldsymbol{b} = [\min\{\underline{\boldsymbol{a}}\,\underline{\boldsymbol{b}},\ \underline{\boldsymbol{a}}\,\overline{\boldsymbol{b}},\ \overline{\boldsymbol{a}}\,\underline{\boldsymbol{b}},\ \overline{\boldsymbol{a}}\,\overline{\boldsymbol{b}}\},\ \max\{\underline{\boldsymbol{a}}\,\underline{\boldsymbol{b}},\ \underline{\boldsymbol{a}}\,\overline{\boldsymbol{b}},\ \overline{\boldsymbol{a}}\,\underline{\boldsymbol{b}},\ \overline{\boldsymbol{a}}\,\overline{\boldsymbol{b}}\}], \tag{1.3}$$

$$\boldsymbol{a}/\boldsymbol{b} = \boldsymbol{a} \cdot [1/\overline{\boldsymbol{b}},\ 1/\underline{\boldsymbol{b}}]. \tag{1.4}$$

Определение 1.1 *Алгебраическая система, носителем которой является множество всех интервалов вещественной оси $\mathbb{IR}$ с определён-*

ными для них интервальными арифметическими операциями сложения, вычитания, умножения и деления, которые определены формулами (1.1)–(1.4)*, называется классической интервальной арифметикой и обозначается* $\mathbb{R}$.

Среди наиболее часто встречающихся операций над интервальными величинами можно выделить следующие функции:

$$\operatorname{mid} \boldsymbol{a} = \frac{1}{2}(\overline{\boldsymbol{a}} + \underline{\boldsymbol{a}}) \quad — \quad \text{середина интервала,} \tag{1.5}$$

$$\operatorname{rad} \boldsymbol{a} = \frac{1}{2}(\overline{\boldsymbol{a}} - \underline{\boldsymbol{a}}) \quad — \quad \text{радиус интервала,} \tag{1.6}$$

$$\operatorname{wid} \boldsymbol{a} = \overline{\boldsymbol{a}} - \underline{\boldsymbol{a}} \quad — \quad \text{ширина интервала,} \tag{1.7}$$

$$\operatorname{opp} \boldsymbol{a} = [-\underline{\boldsymbol{a}}, -\overline{\boldsymbol{a}}] — \text{алгебраически противоположный интервал,} \tag{1.8}$$

$$\operatorname{inv} \boldsymbol{a} = [\overline{\boldsymbol{a}}, \underline{\boldsymbol{a}}] \quad — \quad \text{алгебраически обратный интервал,} \tag{1.9}$$

$$|\boldsymbol{a}| = \max\{|\overline{\boldsymbol{a}}|, |\underline{\boldsymbol{a}}|\} \quad — \quad \text{абсолютное значение интервала,} \tag{1.10}$$

$$\langle \boldsymbol{a} \rangle = \begin{cases} \min\{|\overline{\boldsymbol{a}}|, |\underline{\boldsymbol{a}}|\}, & \text{если } 0 \notin a \\ 0 \quad , & \text{иначе} \end{cases} \quad — \quad \text{мигнитуда интервала.} \tag{1.11}$$

Определение 1.2 *Матрица* $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ *называется M-матрицей, если она представима в виде* $A = sI - P$*, где* P *— неотрицательная матрица* $s > \rho(P)$.

Определение 1.3 *Матрица* $\boldsymbol{A} \in \mathbb{IR}^{n\times n}$ *называется интервальной M-матрицей, если каждая вещественная матрица* $A \in \boldsymbol{A}$ *является M-матрицей.*

Определение 1.4 *Матрица* $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ *называется H-матрицей, если её компрант является M-матрицей.*

Определение 1.5 *Матрица* $\boldsymbol{A} \in \mathbb{IR}^{n\times n}$ *называется интервальной H-матрицей, если каждая вещественная матрица* $A \in \boldsymbol{A}$ *является H-матрицей.*

1.2. Интервальные арифметики

К сожалению алгебраические свойства классической арифметики довольно скудны, по сравнению с наиболее привычными числовыми систе-

мами, таких как поля рациональных чисел $\mathbb{Q}$ или вещественных чисел $\mathbb{R}$, поскольку

1. Всякий невырожденный интервал не имеет обратного элемента относительно арифметических операций;
2. Отсутствует полноценная дистрибутивность умножения относительно сложения и вычитания;
3. Неудовлетворительные порядковые свойства относительно упорядочения по включению.

Из-за отсутствия дистрибутивности невозможно приводить подобные члены, что является серьёзным недостатком интервальной арифметики. Для преодоления этой проблемы необходимо в значительной степени изменить данную арифметику, как алгебраическую систему, что ставит под вопрос целесообразность этого шага. Однако проблему необратимости арифметических операций и плохих порядковых свойств можно частично решить с помощью достроения $\mathbb{IR}$ до некоторой более широкой и полной интервальной арифметики $\mathbb{KR}$, которая была предложена Э. Каухером.

Элементами полной арифметики Каухера $\mathbb{KR}$ являются пары вещественных чисел $[a, b]$ которые не обязательно связаны соотношением $a \leq b$. В результате $\mathbb{KR}$ получается присоединением *неправильных интервалов* к классической интервальной арифметике $\mathbb{IR} = \{[a, b] \,|\, a, b \in \mathbb{R}, a \leq b\}$.

Распространение операций сложения и вычитания на полную интервальную арифметику происходит достаточно просто, а явные формулы для умножения удобно выделить в следующие подмножества, а также представить в виде таблицы 1.1:

$\mathcal{P} := \{\boldsymbol{a} \in \mathbb{KR} \quad | \quad (\underline{\boldsymbol{a}} \geq 0) \,\&\, (\overline{\boldsymbol{a}} \geq 0)\}$ — неотрицательные интервалы,

$\mathcal{Z} := \{\boldsymbol{a} \in \mathbb{KR} \quad | \quad \underline{\boldsymbol{a}} \leq 0 \leq \overline{\boldsymbol{a}}\}$ — нульсодержащие интервалы,

$-\mathcal{P} := \{\boldsymbol{a} \in \mathbb{KR} \quad | \quad -\boldsymbol{a} \in P\}$ — неположительные интервалы,

$\mathrm{dual}\,\mathcal{Z} := \{\boldsymbol{a} \in \mathbb{KR} \quad | \quad \mathtt{dual}\,\boldsymbol{a} \in Z\}$ — интервалы, содержащиеся в нуле.

Также предусмотрены и другие возможности. Например, в указанных арифметиках невозможно делить на нуль содержащие интервалы,

Таблица 1.1 – Умножение в полной интервальной арифметике

$\boldsymbol{a}\cdot\boldsymbol{b}$	$\boldsymbol{b}\in\mathcal{P}$	$\boldsymbol{b}\in\mathcal{Z}$	$\boldsymbol{b}\in-\mathcal{P}$	$\boldsymbol{b}\in\operatorname{dual}\mathcal{Z}$
$\boldsymbol{a}\in\mathcal{P}$	$[\underline{\boldsymbol{a}}\,\underline{\boldsymbol{b}}, \overline{\boldsymbol{a}}\,\overline{\boldsymbol{b}}]$	$[\overline{\boldsymbol{a}}\,\underline{\boldsymbol{b}}, \overline{\boldsymbol{a}}\,\overline{\boldsymbol{b}}]$	$[\overline{\boldsymbol{a}}\,\underline{\boldsymbol{b}}, \underline{\boldsymbol{a}}\,\overline{\boldsymbol{b}}]$	$[\underline{\boldsymbol{a}}\,\underline{\boldsymbol{b}}, \underline{\boldsymbol{a}}\,\overline{\boldsymbol{b}}]$
$\boldsymbol{a}\in\mathcal{Z}$	$[\underline{\boldsymbol{a}}\,\overline{\boldsymbol{b}}, \overline{\boldsymbol{a}}\,\overline{\boldsymbol{b}}]$	$[\min\{\underline{\boldsymbol{a}}\,\overline{\boldsymbol{b}}, \overline{\boldsymbol{a}}\,\underline{\boldsymbol{b}}\},$ $\max\{\underline{\boldsymbol{a}}\,\underline{\boldsymbol{b}}, \overline{\boldsymbol{a}}\,\overline{\boldsymbol{b}}\}]$	$[\overline{\boldsymbol{a}}\,\underline{\boldsymbol{b}}, \underline{\boldsymbol{a}}\,\underline{\boldsymbol{b}}]$	0
$\boldsymbol{a}\in-\mathcal{P}$	$[\underline{\boldsymbol{a}}\,\overline{\boldsymbol{b}}, \overline{\boldsymbol{a}}\,\underline{\boldsymbol{b}}]$	$[\underline{\boldsymbol{a}}\,\overline{\boldsymbol{b}}, \underline{\boldsymbol{a}}\,\underline{\boldsymbol{b}}]$	$[\overline{\boldsymbol{a}}\,\overline{\boldsymbol{b}}, \underline{\boldsymbol{a}}\,\underline{\boldsymbol{b}}]$	$[\overline{\boldsymbol{a}}\,\overline{\boldsymbol{b}}, \overline{\boldsymbol{a}}\,\underline{\boldsymbol{b}}]$
$\boldsymbol{a}\in\operatorname{dual}\mathcal{Z}$	$[\underline{\boldsymbol{a}}\,\underline{\boldsymbol{b}}, \overline{\boldsymbol{a}}\,\underline{\boldsymbol{b}}]$	0	$[\overline{\boldsymbol{a}}\,\overline{\boldsymbol{b}}, \overline{\boldsymbol{a}}\,\underline{\boldsymbol{b}}]$	$[\max\{\underline{\boldsymbol{a}}\,\underline{\boldsymbol{b}}, \overline{\boldsymbol{a}}\,\overline{\boldsymbol{b}}\},$ $\min\{\underline{\boldsymbol{a}}\,\overline{\boldsymbol{b}}, \overline{\boldsymbol{a}}\,\underline{\boldsymbol{b}}\}]$

что значительно ограничивает функционал библиотеки. И встаёт резонный вопрос: "А насколько необходимы столь жёсткие требования?". Ведь действительно, если рассматриваются интервальные величины, в которых лежат истинные значения величин, то вероятность, что в нуль содержащих интервалах это окажется именно нуль ничтожно мала. В связи с этим резонно рассмотреть деление в арифметика Кахана:

$$\boldsymbol{a}/\boldsymbol{b} = \begin{cases} \boldsymbol{a}\cdot[1/\overline{\boldsymbol{b}}, 1/\underline{\boldsymbol{b}}], & \text{если} \quad 0\notin\boldsymbol{b}, \\ ]-\infty,\infty[, & \text{если} \quad 0\in\boldsymbol{a} \text{ и } 0\in\boldsymbol{b}, \\ [\overline{\boldsymbol{a}}/\underline{\boldsymbol{b}}, \infty], & \text{если} \quad \overline{\boldsymbol{a}}<0 \text{ и } \underline{\boldsymbol{b}}<\overline{\boldsymbol{b}}=0, \\ ]-\infty, \overline{\boldsymbol{a}}/\overline{\boldsymbol{b}}]\cup[\overline{\boldsymbol{a}}/\underline{\boldsymbol{b}}, \infty[, & \text{если} \quad \overline{\boldsymbol{a}}<0 \text{ и } \underline{\boldsymbol{b}}<0<\overline{\boldsymbol{b}}, \\ ]-\infty, \overline{\boldsymbol{a}}/\overline{\boldsymbol{b}}], & \text{если} \quad \overline{\boldsymbol{a}}<0 \text{ и } 0=\underline{\boldsymbol{b}}<\overline{\boldsymbol{b}}, \\ ]-\infty, \underline{\boldsymbol{a}}/\underline{\boldsymbol{b}}], & \text{если} \quad 0<\underline{\boldsymbol{a}} \text{ и } \underline{\boldsymbol{b}}<\overline{\boldsymbol{b}}=0, \\ ]-\infty, \underline{\boldsymbol{a}}/\underline{\boldsymbol{b}}]\cup[\underline{\boldsymbol{a}}/\overline{\boldsymbol{b}}, \infty[, & \text{если} \quad 0<\underline{\boldsymbol{a}} \text{ и } \underline{\boldsymbol{b}}<0<\overline{\boldsymbol{b}}, \\ ]\underline{\boldsymbol{a}}/\overline{\boldsymbol{b}}, \infty], & \text{если} \quad 0<\underline{\boldsymbol{a}} \text{ и } 0=\underline{\boldsymbol{b}}<\overline{\boldsymbol{b}}, \\ \varnothing, & \text{если} \quad 0\notin\boldsymbol{a} \text{ и } 0=\boldsymbol{b}. \end{cases}$$

После того, как стало ясно какие операции будут рассматриваться ниже необходимо прийти к пониманию, что же такое *интервальный вектор* и *интервальные уравнения*.

Интервальный вектор определяется как вектор с интервальными компонентами. Его геометрической интерпретацией в пространстве $\mathbb{R}^n$ является прямоугольный параллелепипед, у которого рёбра параллельны координатным осям пространства. Интервальный вектор также называют *брусом*.

Интервальные уравнения или системы уравнений — уравнения или

системы уравнений с параметрами, которые являются интервальными величинами.

В данной работе в бóльшей степени рассматривается интервальная система линейных алгебраических уравнений, которая выглядит следующим образом:

$$\begin{cases} \boldsymbol{a}_{11}x_1 & + & \boldsymbol{a}_{12}x_2 & + & \cdots & + & \boldsymbol{a}_{1n}x_n & = & \boldsymbol{b}_1 \\ \boldsymbol{a}_{21}x_1 & + & \boldsymbol{a}_{22}x_2 & + & \cdots & + & \boldsymbol{a}_{2n}x_n & = & \boldsymbol{b}_2 \\ \vdots & & \vdots & & \ddots & & \vdots & & \vdots \\ \boldsymbol{a}_{m1}x_1 & + & \boldsymbol{a}_{m2}x_2 & + & \cdots & + & \boldsymbol{a}_{mn}x_n & = & \boldsymbol{b}_m \end{cases}$$

Также её можно представить в более коротком и удобном виде:

$$\boldsymbol{A}\,x = \boldsymbol{b} \tag{1.12}$$

где $\boldsymbol{A} = (\boldsymbol{a}_{mn})$ — интервальная $m \times n$ матрица, $x = (x_1, x_2, \ldots, x_n)^\top$, $\boldsymbol{b} = (\boldsymbol{b}_m)$ — интервальный вектор. Однако, стоит отметить, что в некоторых местах будет также рассмотрен более общий нелинейный случай:

$$F(\boldsymbol{a}, x) = \boldsymbol{b}, \tag{1.13}$$

где $F = (F_1(\boldsymbol{a}, x), F_2(\boldsymbol{a}, x), \ldots, F_m(\boldsymbol{a}, x))^\top$, $x = (x_1, x_2, \ldots, x_n)^\top$, $F_i : \mathbb{R}^l \times \mathbb{R}^n \to \mathbb{R}$ — заданные функции, $\boldsymbol{a} = (\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_l), \boldsymbol{b} = (\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_m)$ — интервальные векторы.

Итак, вооружившись основными знаниями можно приступить к непосредственной реализации...

Глава 2

Основные принципы

2.1. Постулаты

При создании любого программного модуля разработчики руководствуются некоторыми основными требованиями. В данном разделе в порядке убывания указаны наиболее приоритетные условия, которыми руководствовались мы.

1. Библиотека должна быть проста в использовании, а написанный код — читаемым и понятным. Хорошо спроектированный модуль, построенный в соответствии с ясной и продуманной структурой, будет проще для изучения и использования. Вряд ли пользователи будут разбираться с беспорядочной структурой, даже если программный инструмент наиболее современен.

2. Библиотека должна удовлетворять общепринятым стандартам — IEEE 754-2008 для арифметики с плавающей точкой на ЭВМ и IEEE 1788-2015 для интервальных вычислений на ЭВМ. По умолчанию пользователь работает в режиме повышенной точности для концов интервалов, в котором количество знаков после запятой — варьируемая величина. Но при желании, если позволяет постановка задачи, пользователь может перейти в режим округления к ближайшему чётному числу, что существенно сократит скорость выполнения алгоритмов.

3. Библиотека должна обеспечивать высокую производительность. Если решение задач полиномиальной сложности будет требовать изрядного количества времени, то вряд ли подобную функциональность можно назвать полезной.

4. Библиотека должна быть кроссплатформенной.[1]

[1]Отметим, что в отличие от других интервальных библиотек, написанных на языке `Python`, `IntvalPy` кроссплатформенна, и это является одним из её главных достоинств.

5. Библиотека должна обеспечивать возможность работы в разных интервальных арифметиках (классическая, Каухера, Кахана и т.д.). Также, в случае, если арифметики совместимы друг с другом, то система должна автоматически переходить от одной к другой.

6. Библиотека должна быть гибкой к изменениям и обеспечивать возможность расширения функциональности. Архитектура библиотеки должна позволять вносить изменения как на низкоуровневой функциональности, так и на верхнеуровневых интервальных методах (решатели уравнений, визуализация множеств решений и т.д.).

2.2. Архитектура библиотеки

Библиотека `IntvalPy`, написанная на языке программирования `Python`, реализует алгебраически замкнутую систему для работы с интервалами, для решения интервальных систем как линейных, так и нелинейных уравнений, а также визуализует множества решений интервальных линейных систем уравнений.

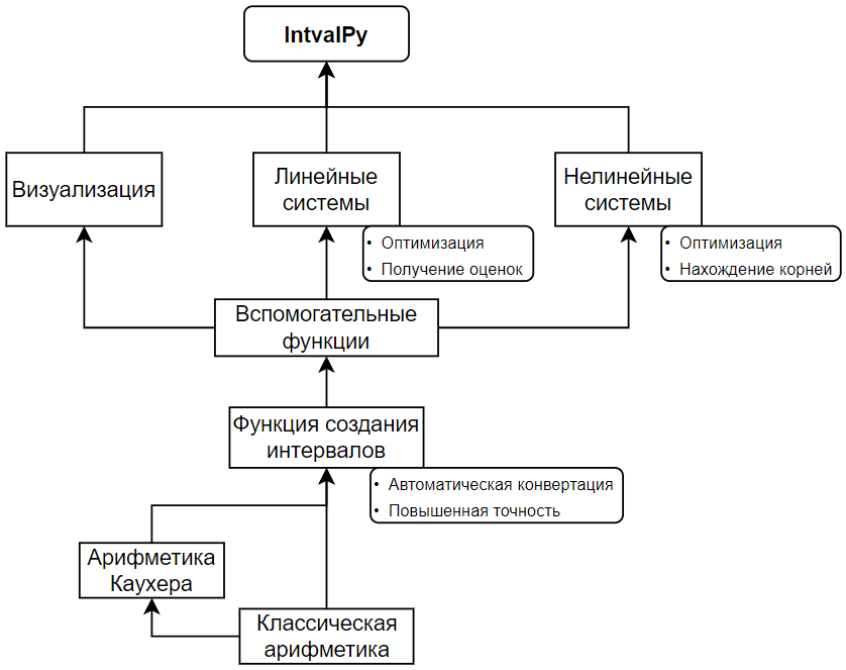
В этом разделе представлены основные идеи, которые определяют архитектуру библиотеки и её реализацию. Предназначенный в большей степени для практических задач модуль по умолчанию хранит концы интервалов в формате `np.float64` (стандартная «двойная точность»), и не использует для каждой операции направленные округления. Это позволяет в значительной степени ускорить вычисления и сократить время работы алгоритмов. Однако в некоторых случаях современных научных вычислений требуется более высокая точность, и поэтому пользователю предоставлена возможность работать с интервалами, концы которых представляются в формате `mpf` («mulpiprecision format», где хранится задаваемое пользователем количество знаков мантиссы). При этом режим округления при арифметических операциях с интервалами на данный момент остаётся стандартным — округление к ближайшему числу с чётной младшей значащей цифрой.

Язык `Python` для реализации нашей библиотеки был выбран потому, что это универсальный и свободно распространяемый язык с открытым исходным кодом, а также кроссплатформенной переносимостью. Кроме того, у языка сформировалось огромное сообщество пользователей и разработчиков, качественная документация и большое количество доступных библиотек, в которых реализованы самые разнообразные инструменты и методы. В частности, в пакете `IntvalPy` для хранения левых и правых концов интервалов (которые могут являться многомерными) активно используется широко распространённый модуль `NumPy`. Его применение особенно полезно потому, что позволяет создавать не только век-

торы, но и матрицы с соответствующими поэлементными операциями. Такой подход в значительной степени ускоряет вычисления, так как сам модуль `NumPy` интегрирован с языком `C++` и именно в нём реализуются циклы.

Реализация библиотеки `IntvalPy` соответствует схеме, показанной на Рис. 2.1, где стрелки обозначают включение (подчинение) отдельных структурных элементов в другие элементы. Наиболее подробно разработаны низкоуровневые операции, а также численные методы для линейных интервальных систем.

Рис. 2.1 – Архитектура IntvalPy



Базовыми классами библиотеки `IntvalPy` являются `Classical` и `Kaucher`, в которых определяются все арифметические операции, абсолютные характеристики интервалов, операции для работы с многомерными интервальными массивами. Также предусмотрены некоторые методы, которые могут менять принадлежность интервала к классу (например, дуализация или получение алгебраически обратного интервала). Для удобства создания интервалов была написана отдельная функция `Interval`, которая определяет к какой арифметике относится интервал или массив интервалов любой размерности. Стоит отметить, что по умолчанию стоит автоматическая конвертация концов интервалов, т. е. интервалы всегда «правильные». Поэтому, если пользователь хочет работать в полной арифметике Каухера, то ему необходимо отключить эту функциональность. Кроме того, в функции `Interval` определяется будут ли данные храниться в формате `np.float64` или `mpf`.

Введение подобной функции соответствует основным принципам, ко-

торые были описаны ранее в §2.1. В целом, библиотека спроектирована как расширяемая, и все её будущие реализации можно будет без особых усилий сделать совместимыми с предшествующими версиями.

Другие библиотеки, использованные при создании `IntvalPy`, также широко известны и могут применяться для других операционных систем. Это соответствует сформулированному выше принципу кроссплатформенности библиотеки.

2.3. Реализация интервальных классов

Одной из первоначальных проблем реализации интервальной библиотеки на Питоне является то обстоятельство, что не существует не только внутренних инструментов для достаточно простой и интуитивной работы с интервальными данными, но и самого типа данных «интервал» попросту нет. Более того, в уже реализованных пользовательских модулях нет возможности работать с интервальными векторами. Это заставляет в каждой новой программе прописывать свои функции, которые будут совершать нужные операции, к примеру, скалярное или матричное умножение. Данная проблема не только создаёт значительные неудобства, но и сильно замедляет исполнение программного кода, поскольку Питон — достаточно медленный интерпретируемый язык. К примеру, C++ работает практически в 50 раз быстрее.

В связи с отмеченными очевидными неудобствами возникает мысль о создании собственного класса Interval, в котором будут устранены описанные недостатки.

Для хранения левых и правых концов интервалов было решено использовать массивы из библиотеки `numpy`, которые могут являться многомерными. Это особенно полезно потому, что можно создавать не только вектора, но и матрицы с соответствующими поэлементными операциями. Для пояснения рассмотрим два точечных массива $a = \{1, 2\}$ и $b = \{3, 4\}$. Тогда для того, чтобы их сложить и получить вектор $c = \{4, 6\}$ достаточно написать $a + b$. Это упрощение не только весьма удобно, но и значительно уменьшает время ожидания для получения ответа. Эта конструкция будет исполняться даже быстрее, так как сама библиотека `numpy` интегрирована с языком `C++` и именно в нём реализуются циклы. Естественно, это происходит с бóльшей скоростью, чем если бы мы писали это непосредственно в Питоне.

После того, как было определено, с помощью какого типа данных будут храниться концы интервала, необходимо понять, как наиболее удобно создать сам объект. Во-первых, будем задавать интервалы указанием

их левых и правых концов (хотя существуют другие способы описания интервалов). Во-вторых, стоит понять, каким образом заносить данные. Как пару чисел, означающих нижний и верхний концы интервала, но одним целым входным параметром, или же разбить на два входных параметра. При создании данного модуля было решено, что нельзя однозначно сказать какой вариант лучше отвечает человеческой интуиции. Поэтому было решено реализовать оба варианта.

Следующая проблема касается корректности введения данных, но уже с точки зрения «правильности» интервалов. Класс `Interval` корректно работает как с правильными интервалами, в которых левый конец всегда меньше либо равен правого, так и с неправильными интервалами. Однако было решено, что пользователь может просто ошибиться и ввести неправильные значения, так что необходимо выполнять проверку правильности вводимых интервалов которая заключается в проведении внутренней конвертации интервала к правильному. Этот подход позволяет более гибко задавать интервалы, к примеру, с помощью генератора случайных чисел для концов интервалов. В таком случае отпадает необходимость каждый раз проводить конвертацию вручную. Однако, если подобный функционал не нужен и возникают ситуации для создания «неправильных» интервалов Каухера, то существует третий входной атрибут `sortQ`, который имеет булев тип и его значение по умолчанию равняется `true`.

Как можно понять из названия атрибута, с помощью него и принимается решение, проводить конвертацию при создании нового интервала или нет. Значение по умолчанию указывает на то, что конвертация проводится, и для создания нового интервала достаточно задать только левые и правые концы.

Практическая пригодность интервального пакета `IntvalPy` обусловлена имеющимися в нём функциональными возможностями. При этом программный модуль продолжает развиваться, в него вводится новый инструментарий.

Операции нижнего функционального слоя модуля `IntvalPy`, доступные для использования, основаны на определении интервальных арифметик. В `IntvalPy` можно выделить классическую интервальную арифметику и полную арифметику Каухера. Однако в некоторых высокоуровневых функциях встречаются выражения с применением деления из арифметики Кэхэна, допускающие нуль в интервале делителя. Таким образом реализованы основные арифметические операции, тригонометрические и математические функции, а также методы для получения абсолютных характеристик интервалов, которые соответствуют стандарту IEEE 1788-2015. В таблице 2.1 резюмированы операции доступные для применения.

Таблица 2.1 – Доступные операции в `IntvalPy`

Операции	Описание
$+$, -, $*$, $/$	Арифметические операции
`pow`, `sqrt`, `abs`, `exp`, `log`	Математические функции
`sin`, `cos`	Тригонометрические функции
`matmul`, `dot`, `transpose`	Матричные операции
$<, \leq, =, \geq, >, \neq, \subset, \subseteq, \cap$	Теоретико-множественные операции
`rad`, `wid`, `mid`, `mig`	Абсолютные характеристики
`dual`, `pro`, `opp`, `inv`	Различные преобразования интервалов

С помощью созданных классов пользователь может конструировать интервалы, работать с интервальными арифметическими операциями, интервальными расширениями элементарных функций, а также применять различные служебные функции — получать характеристики интервалов и т. п. Нахождение числовых характеристик интервалов и различные преобразования интервалов вызываются как соответствующие методы классов.

Важно отметить следующее. Поскольку классы `Classical` и `Kaucher` предоставляют возможность работы с интервальными векторами и матрицами, то необходимо не забывать о том, что аргумент передаётся в функцию «по ссылке» (поверхностная копия), а не «по значению». Поэтому при расширении функциональности библиотеки необходимо создавать глубокие копии аргументов внутри функций (т. е. отдельные объекты), чтобы не столкнуться с неприятными сюрпризами в дальнейшем.

Глава 3

Метод распознающего функционала

Следующим этапом, перед тем, как приступить к решению системы уравнений с интервальными данными необходимо понять, а разрешима ли она. Для этого рассматривается задача о распознавании разрешимости, т.е. непустоты множества решений. В случае интервальной линейной $(m \times n)$-системы уравнений потребуется решить не более чем 2^n линейных неравенств размера $2m + n$. Это следует из факта о выпуклости и многогранности пересечения множеств решений интервальной системы линейных алгебраических уравнений (ИСЛАУ) с каждым из ортантов пространства $\mathbb{R}^n$. Уменьшение количества неравенств принципиально невозможно, что следует из факта труднорешаемости задачи, т.е. её NP-трудности [2]. Ясно, что выше описанный метод применим лишь при малой размерности задачи, поэтому был предложен *метод распознающего функционала.*

3.1. Слабо совместная система

Определение 3.1 *Будем говорить, что интервальная система линейных уравнений* (1.12) *является слабо разрешимой (слабо совместной), если существуют такие точечные значения* $A \in \boldsymbol{A}$ *и* $b \in \boldsymbol{b}$*, что точечная система уравнений* $\boldsymbol{A}\,x = \boldsymbol{b}$ *имеет решение.*

Заметим, что слабая разрешимость интервальной системы уравнений эквивалента непустоте объединённого множества решений (1.11). Впервые такая постановка задачи была поставлена в [4].

Для распознавания разрешимости в линейном случае были придуманы и исследованы в работах [2,6] функционалы $\mathrm{Uni}\,(x, \boldsymbol{A}, \boldsymbol{b})$ и $\mathrm{Uss}\,(x, \boldsymbol{A}, \boldsymbol{b})$,

которые выглядят следующим образом:

$$\operatorname{Uni}(x, \boldsymbol{A}, \boldsymbol{b}) = \min_{1 \le i \le m} \left\{ \operatorname{rad} b_i - \left\langle \operatorname{mid} b_i - \sum_{j=1}^{n} \boldsymbol{a}_{ij} x_j \right\rangle \right\},$$

$$\operatorname{Uss}(x, \boldsymbol{A}, \boldsymbol{b}) = \min_{1 \le i \le m} \left\{ \operatorname{rad} b_i + \sum_{j=1}^{n} (\operatorname{rad} a_{ij}) |x_j| - \left| \operatorname{mid} b_i - \sum_{j=1}^{n} (\operatorname{mid} a_{ij}) x_j \right| \right\},$$

где $\boldsymbol{A} = (\boldsymbol{a}_{ij})$ — интервальная $m \times n$ матрица, $\boldsymbol{b} = (\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_m)$ — интервальный m-вектор, $x = (x_1, x_2, \ldots, x_n)^\top$.

В случаях, когда интервальные переменные фиксированы и необходимо подчеркнуть, что функционалы рассматриваются исключительно в зависимости от переменной x, будем писать $\operatorname{Uni}(x)$, $\operatorname{Uss}(x)$, опуская упоминание о зависимости от интервальных векторов $\boldsymbol{A}$ и $\boldsymbol{b}$. Своё название распознающие функционалы получили в силу того, что они посредством своего знака «распознают» принадлежность своего аргумента множеству Ξ_{uni}.

Очень важным отличием между этими двумя функционалами является то, что $\operatorname{Uni}(x)$ достигает своего конечного максимума по x на всём пространстве $\mathbb{R}^n$, а $\operatorname{Uss}(x)$ может обращаться в бесконечность. Однако если множество решений ограничено, то функционал $\operatorname{Uss}(x)$ ограничен сверху и достигает конечного глобального максимума на всём пространстве.

3.2. Сильно совместная система

Определение 3.2 *Будем говорить, что интервальная система линейных уравнений* (1.12) *является сильно разрешимой (сильно совместной), если для любых $A \in \boldsymbol{A}$ найдётся такое $b \in \boldsymbol{b}$, что точечная система уравнений $\boldsymbol{A}\, x = \boldsymbol{b}$ имеет решение.*

Для пояснения содержательного смысла допускового множества решений рассмотрим заданную в виде «чёрного ящика» интервальную систему в нелинейной постановке. Дело в том, что необходимо различать те или иные случаи прохождения графика восстанавливаемой зависимости через брусы в силу того, что экзогенные и эндогенные переменные системы (соответствующие независимым аргументам функции и зависимым переменным) различны по своему функциональному назначению и

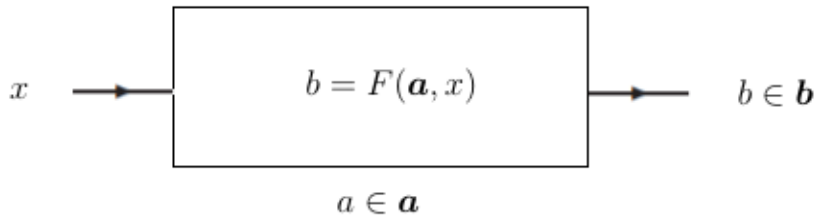


Рис. 3.1 – Модель для интерпретации допускового множества решений

их замеры могут выполняться отличными друг от друга способами и, быть может, в разное время.

Мы хотим понять, существуют ли такие входные сигналы $\widetilde{x}$, что при любых возможных реализациях параметров a из $\boldsymbol{a}$ мы всё же получим на выходе отклик $b \in \boldsymbol{b}$. Допусковое множество решений Ξ_{tol} как раз совпадает со множеством всех таких $\widetilde{x} \in \mathbb{R}^n$. Также это множество является «наиболее устойчивым» из всех множеств AE-решений [3].

Вернёмся для начала к линейному случаю и будем находиться в условиях естественного интервального расширения.

Предложение 3.2.1 *Выражением*

$$\operatorname{Tol}(x, \boldsymbol{A}, \boldsymbol{b}) = \min_{i\,=\,1,\ldots,m} \left\{ \operatorname{rad} b_i \; - \; \left| \operatorname{mid} b_i - \sum_{j=1}^{n} \boldsymbol{a}_{ij} x_j \right| \right\} \tag{3.1}$$

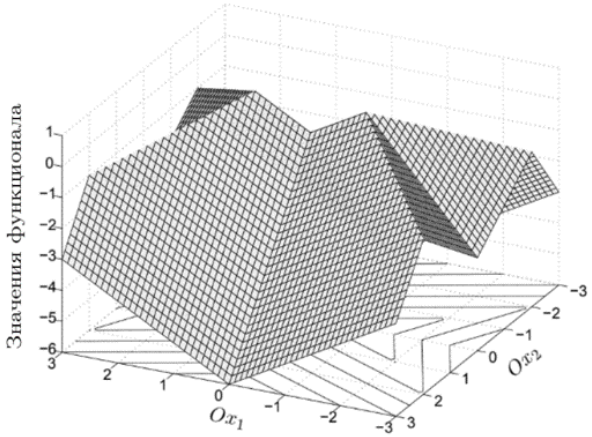
задаётся отображение $\operatorname{Tol} : \mathbb{R}^n \times \mathbb{R}^l \times \mathbb{R}^m \to \mathbb{R}$*, такое что если точка* $x \in \mathbb{R}^n$ *принадлежит допусковому множеству решений* Ξ_{tol} *интервальной системы уравнений* $\boldsymbol{A}\,x = \boldsymbol{b}$*, то отображение* Tol *неотрицательно в* x*, т.е.*

$$x \in \Xi_{tol} \qquad \Rightarrow \qquad \operatorname{Tol}(x, \boldsymbol{a}, \boldsymbol{b}) \geq 0.$$

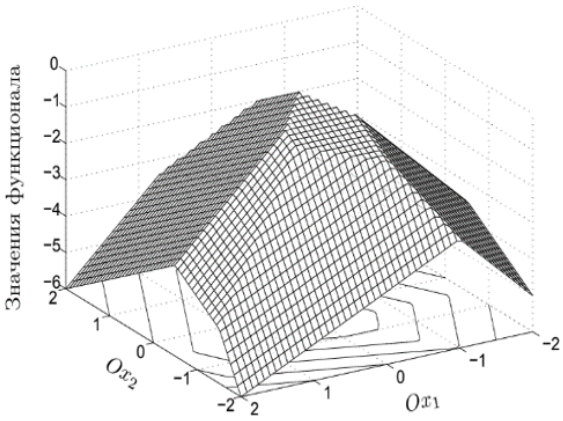
Свойства введённого функционала схожи с его предшественниками, к примеру, $\operatorname{Tol}(x)$ также является полиэдральным, однако наблюдается существенное отличие, а точнее:

Предложение 3.2.2 *Функционал* Tol *— вогнутый по* x *в области* $\mathbb{R}^n$*.*

На графиках ниже, взятых из работы [5], показано значение функционала для линейного случая в условиях интервальных и точечных входных данных.



Многоэкстремальный случай



Унимодальная функция

3.3. Парадокс интервального оценивания

Рассмотрим парадокс интервального оценивания Демиденко Е.З. [7], который описан фразой «чем лучше, тем хуже» и заключается в том, что чем выше точность исходных данных, тем хуже для оценивания параметров и наоборот, при более неточных данных можно получить более богатый набор результатов.

Один из способов преодоления этого парадокса, описанный в [8], заключается в том, что интервальные неопределённости данных не до конца описывают всевозможные значения величин. Из-за этого может быть так, что выбранная модель не имеет полного согласования с эмпирическими данными. А это ведёт к необходимости решению задачи минимизации этого несогласования.

Данный способ особенно востребован в тех случаях, когда необходимо рассматривать конкретно заданную модель, a priori считая, что она является лучшей.

Теперь, решив идти по данному способу преодоления проблемы, нужно количественно показать несогласованность данных введя «меру несогласования». Сделав этот шаг искомой оценкой, к примеру, будет являться та точка пространства, в которой достигается минимальная несогласованность.

Из выше изложенных параграфов можно понять, что на роль «меры согласования/несогласования» данных можно взять любой из распознающих функционалов. Действительно, в случае, когда рассматривается точная область значений, а не её естественное интервальное расширение, функционалы удовлетворяют весьма естественным требованиям для этого. К примеру, при непустом информационном множестве принимают положительные значения для точек, лежащих в этом множестве.

Таким образом для того, чтобы понять разрешима ли система необхо-

димо провести глобальную оптимизацию функционала и посмотреть на знак максимума max.

- Если $max \geq 0$, то аргумент доставляющий максимум функции лежит в непустом множестве параметров, согласующихся с данными;
- Если $max < 0$, то множество параметров, согласующихся с данными, пусто, но этот аргумент минимизирует несогласованность.

В случаях, когда значение максимума отрицательно, то можно воспользоваться коррекцией интервальной системы:

$$\max_x \operatorname{Uni}(x, \boldsymbol{a}, \boldsymbol{b} + C \cdot \boldsymbol{e}) = \max_x \operatorname{Uni}(x, \boldsymbol{a}, \boldsymbol{b}) + C,$$

где $\boldsymbol{e} = ([-1, 1], \ldots, [-1, 1])^\top$, $C \geq 0$ — константа. Таким образом, если модель не теряет своей адекватности, можно расширять вектор правой части и получить согласующиеся данные. Подробное обоснование Вы можете найти в работах [6, 8].

3.4. Интервальная оптимизация

В этом параграфе рассмотрим применение интервальных методов глобальной оптимизации для поиска глобального максимума распознающих функционалов.

Из предложения 3.2.2 следует, что для линейных задач функционал Tol является унимодальной функцией. Однако в нелинейных постановках этот факт не всегда имеет место быть. Например, если рассматривается экспоненциальная модельная функция, то нет никаких гарантий, что при оптимизации, которая применяется в линейном случае, будет достигнут глобальный максимум. В тоже время существует много прикладных задач которые уходят за пределы линейных постановок, а значит необходим инструмент с помощью которого можно достоверно найти глобальный максимум распознающего функционала. Для преодоления этой проблемы сконструируем интервальный метод глобальной оптимизации.

Первым шагом является получение интервального расширения целевой функции. На этом этапе возникает проблема, что входные значения $a \in \boldsymbol{a}$ могут входить в модельную функцию $F(\boldsymbol{a}, x)$ более одного раза, а значит интервальное расширение будет получаться более широким. Другими словами не факт, что всякое значение из интервального расширения у таких функций может действительно реализоваться. В качестве примера предлагается рассмотреть следующую функцию:

$$\boldsymbol{F}(\boldsymbol{a}, x_1, x_2) = \boldsymbol{a} \cdot x_1^2 + x_2^3/\boldsymbol{a}. \tag{3.2}$$

Если $a \in \boldsymbol{a} = [1, 2]$, $x_1 = 1, x_2 = 2$, то $\boldsymbol{F}(\boldsymbol{a}, x_1, x_2) = [5, 10]$. Однако легко проверить, что $\forall a \in \boldsymbol{a}$ значения функции F не выходят за пределы интервала $[6, 9]$. Значит во время оптимизации может наступить такой момент, когда дробление бруса $\boldsymbol{x}$ не будет приносить никакой пользы, т. к. интервальное расширение не будет уменьшаться. Это может привести к получению неправильной точки оптимума. Для преодоления этой проблемы можно попробовать видоизменить модельную функцию так, чтобы входные параметры встречались не более одного раза. Такой подход может несколько увеличить количество рассматриваемых функций, но в целом мало применим. Поэтому следует заметить, что подобного эффекта можно добиться, если входные данные заданы как точные величины. Этот частный случай очень важен, т. к. очень часто встречается в практических важных задачах. Таким образом в дальнейшем будем рассматривать только те модельные функции, которые удовлетворяют условиям выше, чтобы получить как можно более узкое интервальное расширение.

В результате мы можем явным образом указать целевую функцию которую будем оптимизировать. Без ограничения общности рассмотрим распознающий функционал Tol:

$$\mathbf{Tol}\,(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b}) = \min_{i\,=\,1,\ldots,m} \Big\{ \operatorname{rad} \boldsymbol{b}_i \;-\; \Big| \operatorname{mid} \boldsymbol{b}_i - \boldsymbol{F}(\boldsymbol{a}, \boldsymbol{x}) \Big| \Big\}. \tag{3.3}$$

Из теории следует, что чем меньше ширина бруса по которому производится оценивание, тем меньше абсолютная погрешность интервального оценивания области значения функции. Значит мы можем построить простейшую процедуру для уточнения внешней оценки целевой функции. Наиболее подробно можно ознакомиться из учебных материалов по интервальному анализу, например в [2, 14]. Однако применение такого алгоритма едва ли можно назвать успешным для серьёзных задач. Основной упор делается на бисекцию интервалов, и при увеличении размерности задачи эффективность подхода всё менее ощутима. Поэтому вторым шагом модифицируем процесс оптимизации.

Во-первых, кроме оценивания целевой функции по целым брусам будем вычислять её значения в каких-то точках этих этих брусов. Полученное значение является нижней границей истинного глобального максимума, а значит появляется возможность чистить рабочий список от записей, которые заведомо не содержат нужную точку оптимума.

Во-вторых, выявление монотонности целевой функции на брусах из

рабочего списка позволяет добиться уменьшения размерности этих брусов.

Указанные модификации позволяют в значительной степени сократить количество итераций для достижения точки оптимума. Однако для работы с прикладными задачами всё также малопригоден. Поэтому требуется дальнейшее развитие данного подхода.

3.5. Примеры

Ниже приведены примеры работы распознающих функционалов Uni и Tol для интервальных систем алгебраических уравнений в линейном и нелинейном случаях.

Для нахождения глобального максимума распознающих функционалов были реализованы интервальные методы оптимизации, а также была переписана свободно распространяемая программа `tolsolvty`. Кроме этого, алгоритм был модифицирован для более быстрого выполнения процесса оптимизации и обобщён на нелинейный случай. Для функционала Uni, если рассматривать оптимизацию в точечном смысле, используется алгоритм негладкой оптимизации ralgb5, который разработан П.И. Стецюком [15].

Рассмотрим применение функционалов в линейном случае. В качестве тестовой системы предлагается рассмотреть систему Ноймайера-Райхмана:

Листинг 3.1 – Оптимизация распознающих функционалов в линейном случае

```
1  import intvalpy as ip
2
3  A, b = ip.Neumeier(6, 3.5)
4  tol = ip.linear.Tol(A, b, maxQ=True)
5  uni = ip.linear.Uni(A, b, maxQ=True, x0=[1, 2,
       3, 4, 5, 6])
6  print('tol: ', tol)
7  print('uni: ', uni)
8  # tol:  (True, array([0., 0., 0., 0., 0., 0.]),
       1.0)
9  # uni:  (True, array([1., 2., 3., 4., 5., 5.]),
       -16.5)
```

Из примера видно, что максимум функционала Uni отрицательное значение. Однако такого не может быть, т. к. если система сильно согласована, то тем более существует объединённое множество решений. Это произошло из-за того, что процесс оптимизации завершился в каком-то локальном экстремуме. Таким образом пример демонстрирует зависи-

мость выбора начального приближения у функционала Uni, если используется не интервальная оптимизация.

В качестве следующего примера рассмотрим поиск глобального максимума у распознающего функционала Tol в нелинейном случае с модельной функцией 3.2. Для этого создадим точные входные данные a, вычислим выходные значения $b = F(a, x)$ и зададим им некоторую погрешность. После чего применим методы интервальной оптимизации.

Листинг 3.2 – Интервальная оптимизация функционала Tol в нелинейном случае

```
1   import intvalpy as ip
2   import numpy as np
3
4   f = lambda a, x: a*x[0]**2 + x[1]**3 / a
5
6   dx1 = lambda a, x: 2*a*x[0]
7   dx2 = lambda a, x: 3*x[1]**2 / a
8   grad = np.array([dx1, dx2])
9
10  a = np.array([1.26549508, 1.31415286,
        1.40347574, 1.49459073, 1.85887147])
11  b = ip.Interval([
12            [-0.00177322, 0.000847354],
13            [-0.00183278, 0.0007878],
14            [-0.000424848, 0.00219573],
15            [-0.000662522, 0.00195806],
16            [-0.00175955, 0.000861029]
17        ])
18
19  x0 = ip.Interval([[-1, 1], [-1, 1]])
20  _, xx, ff = ip.nonlinear.Tol(f, a, b, maxQ=True,
        grad=grad, x0=x0, maxiter=10000)
21  # xx = Interval([[-0.0115783, -0.0115783],
        [-0.00974234, -0.00974234]])
22  # ff = Interval(0.000612334, 0.000612334)
```

Глава 4

Метод граничных интервалов

4.1. Мотивация

Методом граничных интервалов называется метод исследования и визуализации полиэдральных множеств в Евклидовых пространствах $\mathbb{R}^2$ и $\mathbb{R}^3$ основанный на вычислении и использовании специальной `матрицы граничных интервалов` для систем линейных неравенств. Указанный метод был предложен И.А. Шарой и подробно описан в [12, 13]. Кроме того, его реализацию можно найти в системе `MATLAB`, являющейся платной.

При решении задач нередко возникает необходимость визуализации решений систем линейных неравенств, а также интервальных линейных систем уравнений. Например, это необходимо для наглядных средств обучения в процессе изучения тем «Линейные неравенства», «Линейное программирование» и др. Кроме того, подобный функционал может быть полезен при поступлении обращений от пользователей различных систем по вопросам поддержки принятия решений.

Для удовлетворения подобных запросов можно прибегнуть к методам решения проблемы перечисления вершин. Однако существующие реализации имеют ряд недостатков: работа только с квадратными системами, плохая обработка неограниченных и «тощих» множеств. Также значительная часть программных реализаций являются платными. Целью данной главы является реализация *метода граничных интервалов*, который предназначен для исследования и визуализации полиэдральных множеств, а также лишён вышеописанных недостатков.

Для общего понимания работы алгоритма укажем его основные шаги:

1. Формирование матрицы граничных интервалов;

2. Изменение матрицы граничных интервалов с учётом окна отрисовки;

3. Построение упорядоченных вершин полиэдрального множества решений;

4. Вывод построенных вершин и (если надо) отрисовка полиэдра.

4.2. Основы метода

В данном разделе приведено краткое описание *метода граничных интервалов*.

Пусть задана система линейных неравенств

$$A\,x \geq b, \qquad A \in \mathbb{R}^{n\times m},\ x \in \mathbb{R}^n,\ b \in \mathbb{R}^m \tag{4.1}$$

относительно неизвестного x. Множество решений этой системы обозначим через H, k-ю строку матрицы A через A_k, а множество решений k-ой строки системы неравенств через H_k.

Ясно, что множество решений H системы (4.1) получается путём пересечения множеств решений каждой строки H_k. А его граница ∂H состоит граничных точек, которые принадлежат границе хотя бы одного из ∂H_k:

$$\partial H = H \cap (\bigcup_k \partial H_k) = \bigcup_k (H \cap \partial H_k). \tag{4.2}$$

Если k-е неравенство $A_k\,x \geq b_k$ граничное, то пересечение гиперплоскости $A_k\,x = b_k$ с множеством H называется *опорой* k-го неравенства и обозначается S_k.

Таким образом, если мы введём множество индексов граничных неравенств I_b, то из (4.2) получим:

$$\partial H = \bigcup_{k\in I_b} S_k, \tag{4.3}$$

т. е. опоры граничных неравенств образуют границу множества решений (4.1).

Определим понятие *граничного интервала*. Оно вводится для системы линейных неравенств и включает в себя три составляющих: *индекс*, *опора* и *направление*.

Индекс граничного интервала связывает интервал с неравенством из системы (4.1). Всякое граничное неравенство системы порождает один граничный интервал;

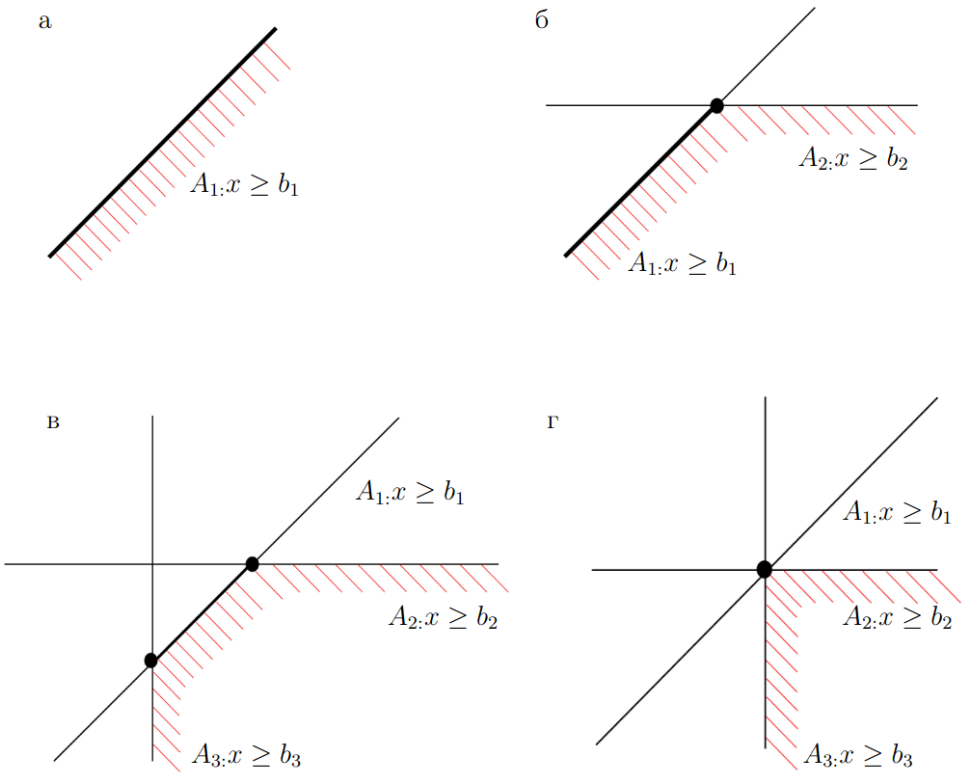


Рис. 4.1 – Типы опор граничного интервала: а) прямая б) луч в) отрезок г) точка.

Опора представляет граничный интервал как множество точек плоскости. *Опорой граничного интервала* с индексом k будем называть опору k-го неравенства системы (4.1). Таким образом опора выпуклое и замкнутое подмножество прямой, а поскольку опора не может быть пустым множеством, то опоры граничных интервалов делятся на четыре типа: прямая, луч, отрезок, точка. На рис. 4.1 приведены примеры;

Направление граничного интервала задаёт движение по прямой $A_k\, x = b_k$, такое что полуплоскость $A_k\, x \geq b_k$ остаётся справа. Множество решений H удовлетворяет неравенству $A_k\, x \geq b_k$, а значит целиком лежит в полуплоскости, задаваемой этим неравенством.

Поэтому можно считать, что при движении по прямой в направлении граничного интервала с индексом k, множество решений H тоже остаётся справа. Численное представление направления граничного интервала, в двумерном случае, можно задать вектором $(-A_{k2}, A_{k1})^\top$.

Рассмотрим вычисление граничного интервала для двумерного случае. Для ненулевой строки A_k, введём внутреннюю систему координат на прямой $A_k\, x = b_k$.

- Началом координат $\overline{O}$ внутренней системы берём проекцию точки $(0, 0)$ исходной системы координат на прямую $A_k\, x = b_k$. С помощью преобразований получаем $\overline{O} = b_k A_k^\top / \|A_k\|_2^2$, где $\|A_k\|_2$ — 2-норма вектора.

- В качестве единичного вектора внутренней системы координат берём вектор направления на прямой $A_k\,x = b_k$.
- Введём обозначение внутренней координаты y.

Таким образом мы можем получить параметрическую запись прямой $A_k\,x = b_k$:

$$\frac{b_k}{\|A_k\|_2^2}A_k^\top + (-A_{k2}, A_{k,1})^\top y, \qquad y \in \mathbb{R}.$$

Найдём пересечение множества решений $A\,x \geq b$ с этой прямой. Для этого выполним замену:

$$x \to \frac{b_k}{\|A_k\|_2^2}A_k^\top + (-A_{k2}, A_{k,1})^\top y.$$

После замены получим систему линейных неравенств с одним неизвестным y, которую можно легко решить. Если множество решений не пусто, то неравенство $A_k\,x \geq b_k$ порождает граничный интервал.

4.3. Примеры

Ниже приведены примеры работы пакетов `lineqs`, `IntLinIncR2` и `IntLinIncR3` для систем линейных неравенств и интервальных систем линейных алгебраических уравнений с различными типами множеств решений в двумерном, а также в трёхмерном случаях.

Сначала рассмотрим двумерный случай, который соответствует Рис. 4.2. Для этого создадим тестовую интервальную систему предложенную С.П. Шарым. Поскольку система интервальная, то мы можем рассмотреть объединённое и допусковое множества решений. Система хорошо изучена, поэтому будет легко убедиться в правильности полученного ответа.

$$\begin{pmatrix} [1,2] & [-0.77, 0.65] \\ [-0.77, 0.65] & [1,2] \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} [-1,1] \\ [-1,1] \end{pmatrix} \tag{4.4}$$

Кроме интервальной системы также предлагается точечную систему линейных неравенств, множество решений которого является полиэдр с двенадцатью вершинами.

$$\begin{pmatrix} 3 & 1 \\ 2 & 2 \\ 1 & 3 \\ -1 & 3 \\ -2 & 2 \\ -3 & 1 \\ -3 & -1 \\ -2 & -2 \\ -1 & -3 \\ 1 & -3 \\ 2 & -2 \\ 3 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \geq \begin{pmatrix} 18 \\ 16 \\ 18 \\ 18 \\ 16 \\ 18 \\ 18 \\ 16 \\ 18 \\ 18 \\ 16 \\ 18 \end{pmatrix} \tag{4.5}$$

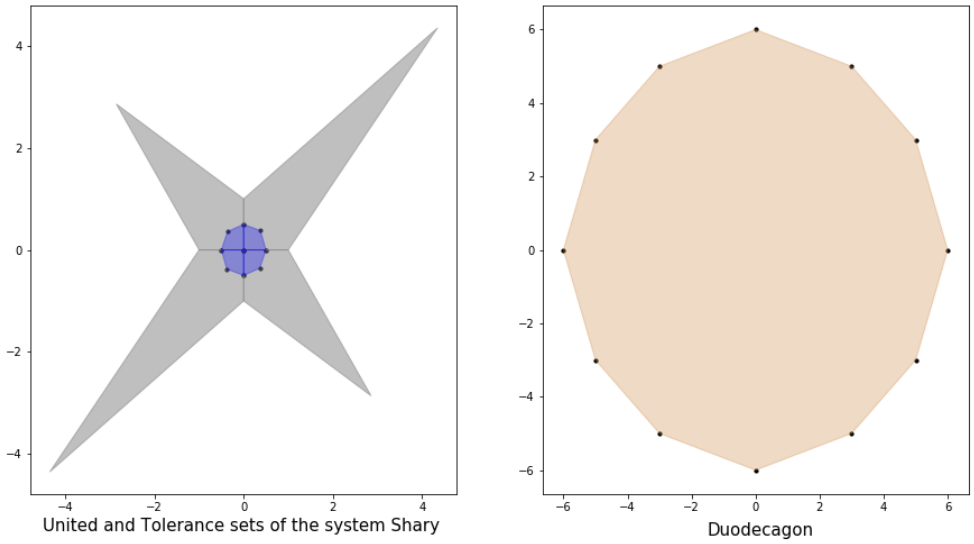


Рис. 4.2 – Множества решений систем 4.4 и 4.5.

В качестве следующего примера предлагается рассмотреть трёхмерный случай с неограниченными и «тощими» множествами решений. Для этого визуализируем объединённое множество решений у систем 4.6 и 4.7:

$$\begin{pmatrix} [-1,1] & [-2,2] & [-2,2] \\ [-2,2] & [-1,1] & [-2,2] \\ [-2,2] & [-2,2] & [-1,1] \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} [2,2] \\ [2,2] \\ [2,2] \end{pmatrix} \tag{4.6}$$

$$\begin{pmatrix} [-1,1] & [0,0] & [0,0] \\ [0,0] & [-1,1] & [0,0] \\ [0,0] & [0,0] & [-1,1] \\ [1,1] & [0,0] & [0,0] \\ [0,0] & [1,1] & [0,0] \\ [0,0] & [0,0] & [1,1] \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} [1,1] \\ [1,1] \\ [1,1] \\ [-1,2] \\ [-1,2] \\ [-1,2] \end{pmatrix} \tag{4.7}$$

В случае, если множество решений неограниченно, то алгоритм самостоятельно выберет границы отрисовки. Однако пользователь сам может указать их явным образом. Для понимания, что множество решений обрезано, плоскости окрашиваются в красный цвет.

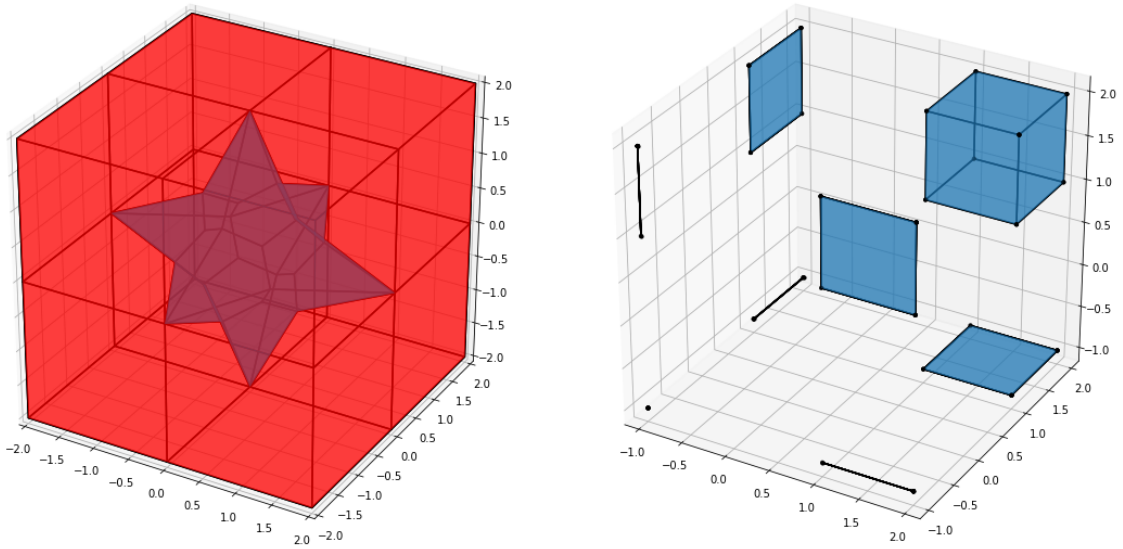


Рис. 4.3 – Объединённые множества решений интервальных линейных систем 4.6 и 4.7.

На основе вышеописанного функционала можно сделать «шаг дальше» и реализовать визуализацию N-мерного пространства путём среза N-мерной фигуры и посмотреть, как выглядит набор решений с фиксированными N-3 (N-2) параметрами. Для наибольшей наглядности стоит рассматривать гиф изображения, где фиксированная переменная меняется с течением времени. Однако если переменных слишком много, то наиболее удобным инструментом являются «ползунки» с помощью которых можно динамически менять значения фиксированных переменных.

Глава 5

Внешнее оценивание

Одна из отличительных особенностей интервальной библиотеки `IntvalPy` — реализация в ней высокоуровневых решателей для различных задач интервального анализа и его приложений. В частности, в библиотеке имеются развитые методы решения как квадратных, так и переопределённых систем интервальных линейных уравнений. Очень популярны, в частности, внешние оценки множеств решений, с помощью объемлющих интервальных брусов. Для их получения предложены интервальные методы Гаусса, Гаусса-Зейделя, процедура Хансена-Блика-Рона, метод Рона, методы дробления решений (PSS-методы), методы дробления параметров (PPS-методы) и др.

В данной главе будут рассмотрены методы для внешнего оценивания объединённого множества решений. Это множество решений является, по-видимому, наиболее популярным из различных множеств решений для интервальных систем уравнений. Его пустота/непустота может быть исследована с помощью распознающего функционала, называемого Uni, но в общем случае эта задача является труднорешаемой (NP-трудной), а распознающий функционал Uni в общем случае является многоэкстремальным.

5.1. Квадратные системы

В качестве первого шага приведём широко известную характеризацию Бекка, которая активно используется при исследовании и получении оценок у объединённого множества решений ИСЛАУ.

Характеризация Бекка: Пусть заданы $\boldsymbol{A} \in \mathbb{IR}^{m\times n}$ и $\boldsymbol{b} \in \mathbb{IR}^m$. Вектор $x \in \mathbb{R}^n$ принадлежит множеству решений $\Xi_{uni}(\boldsymbol{A}, \boldsymbol{b})$ интервальной системы линейных алгебраических уравнений $\boldsymbol{A}x = \boldsymbol{b}$ тогда и только тогда,

когда выполнено любое из условий

$$\boldsymbol{A} \cdot x \cap \boldsymbol{b} \neq \varnothing \qquad\qquad 0 \in \boldsymbol{A} \cdot x - \boldsymbol{b}$$

Если интервальная матрица $\boldsymbol{A}$ имеет специальную форму (например, верхняя треугольная), тогда оптимальная внешняя оценка может быть получена путём последовательного оценивания компонент, где в уравнении присутствует одна неизвестная переменная. Такой подход называется *обратной подстановкой* для верхней треугольной матрицы.

Эта идея активно используется в методе исключения Гаусса, который является популярнейшим алгоритмом вычислительной линейной алгебры. Поэтому создатели библиотеки решили добавить интервальную версию этого алгоритма в программный модуль `IntvalPy`.

В качестве очевидных плюсов можно отметить, что алгоритм прост в своей реализации, и не требует большого количества времени для своего завершения. Однако сразу возникает вопрос: "Каково качество внешних оценок полученных данным алгоритмом?". Поскольку интервальный метод Гаусса является простым интервальным расширением его точечного метода, то исходя из известных теорем следует, что достигается только первый порядок точности. Таким образом, если неточность во входных и выходных параметрах относительно малы, то оценка объединённого множества решений также будет неплохой. Однако, если ширина элементов в ИСЛАУ достаточно высока, то результат, даваемый алгоритмом, может быть неудовлетворительным.

Следующий алгоритм который следует рассмотреть, по-видимому, является метод Гаусса-Зейделя. Это итерационная процедура предназначенная для сужения уже известной оценки параметров. Например, пользователь может задать начальный брус в котором точно находится решение. Если же брус был выбран так, что он не пересекает множество решений, то начиная с некоторого момента алгоритм выдаст ошибку сообщая об этой проблеме.

Ясно, что финальным результатом данной процедуры является брус $\boldsymbol{X}$, не более широкий, чем его начальное приближение $\boldsymbol{X_0}$. Однако хотелось бы заранее знать, когда $\boldsymbol{X}$ действительно уже, чем $\boldsymbol{X_0}$. В случае, если матрица ИСЛАУ является М-матрицей или Н-матрицей, то можно опираться на следующую теорему:

Теорема 5.1.1 *Если вектор $\boldsymbol{X}^* \in \mathbb{IR}^n$ является пределом последовательности, порождаемой интервальным методом Гаусса-Зейделя в применении к интервальной линейной системы $\boldsymbol{A}\, x = \boldsymbol{b}$, то*

$$\langle \boldsymbol{A} \rangle |\boldsymbol{X}^*| \leq |\boldsymbol{b}|.$$

Если же $\boldsymbol{A}$ *— интервальная H-матрица, то*

$$|\boldsymbol{X}^*| \leq \langle \boldsymbol{A} \rangle^{-1} |\boldsymbol{b}|.$$

Кроме того, А. Ноймайер показал, что если матрица $\boldsymbol{A}$ не является H-матрицей, то улучшение любой начальной оценки не происходит. Таким образом этот фактор ограничивает применимость данного алгоритма. Однако, если процедура имеет место быть, то зачастую можно получить оптимальные внешние оценки объединённого множества решений.

5.2. Переопределённые системы

В параграфе выше были рассмотрены методы для внешнего оценивания квадратных систем. Однако, зачастую на практике измерений больше чем переменных, что приводит к необходимости решать переопределённые системы. Для этих целей был реализован метод Рона [17], а также PSS-методы [2].

Метод, предложенный Дж. Роном в статье [17], для получения вектора-решения, основан на решении вспомогательного квадратного линейного неравенства. Для получения данного неравенства активно используется наиболее представительная точечная матрица A_c из интервальной матрицы $\boldsymbol{A}$, т.е. $A_c = mid(\boldsymbol{A})$. Реализованный алгоритм является простейшей вариацией алгоритма предложенного в статье и не даёт оптимальное оценивание множества решений.

Алгоритм для оптимального внешнего оценивания объединённого множества решений не был в реализован в силу его трудоёмкости. Кроме того, указанная процедура является *финально гарантирующей*. Это означает, что для получения бруса-решения, в котором достоверно содержится множество решений, необходимо, чтобы алгоритм успешно завершился. Однако, в силу того, что задача NP-трудна, пользователи не всегда могут довольствоваться этой "роскошью". Более того, в прикладных задачах зачастую имеется большое количество переменных, а также измерений, что ещё больше усугубляет положение дел.

Для преодоления этой проблемы были предложены методы дробления решений (PSS-методы) [2], которые являются *последовательно гарантирующими* и предназначены для нахождения внешних оптимальных оценок множеств решений интервальных систем линейных алгебраических уравнений (ИСЛАУ) $\boldsymbol{A}x = \boldsymbol{b}$. Термин *последовательно гарантирующий* означает, что, например, при достижении максимального количества итераций и соответствующей остановки алгоритма вектор-решение будет достоверно содержать множество решений (при условии, что оно нахо-

дилось в начальном брусе). В качестве базового метода внешнего оценивания в программе используется интервальный метод Гаусса (функция `Gauss`), если система является квадратной. В случае, если система переопределённая, то применяется простейший алгоритм, предложенный Дж. Роном (функция `Rohn`). Кроме этого, если оценивать все компоненты нет необходимости, то можно оценить одну любую ν-ю компоненту, что также является плюсом данного подхода.

5.3. Примеры

В Листинге 5.1 представлено численное решение модельной тестовой задачи С.П. Шарого [16] с размерностью 20×20. Поскольку матрица в этой тестовой системе является H-матрицей, то интервальный метод Гаусса применим и выдаёт внешнюю оценку объединённого множества решений.

Листинг 5.1 – Интервальный метод Гаусса

```
import intvalpy as ip

A, b = ip.Shary(20)
gauss = ip.linear.Gauss(A, b)
```

Для следующего примера рассмотрим систему 4.7 и попробуем найти оптимальную внешнюю оценку. Для это воспользуемся реализованными PSS-методами, которые могут решать переопределённые системы. Напомним, что задача вычисления оптимальных или гарантированно близких к оптимальным оценок объединённого множества решений ИЛСАУ является NP-трудной, поэтому в библиотеке `IntvalPy` реализована гибридная версия PSS-методов, удобная для решения практических задач значительной размерности.

Листинг 5.2 – Метод дроблений решений

```
import intvalpy as ip
ip.precision.extendedPrecisionQ = False

A = ip.Interval([
    [[-1, 1], [0, 0], [0, 0]],
    [[0, 0], [-1, 1], [0, 0]],
    [[0, 0], [0, 0], [-1, 1]],
    [[1, 1], [0, 0], [0, 0]],
    [[0, 0], [1, 1], [0, 0]],
    [[0, 0], [0, 0], [1, 1]]
])
b = ip.Interval([[1, 1], [1, 1], [1, 1], [-1,
   2], [-1, 2], [-1, 2]])
pss = ip.linear.PSS(A, b)
```

В качестве результата мы получим классические интервалы, не смотря на то, что само множество решений распадается на куски (т.е. не является связным). Поскольку алгоритм завершается успешно, то можно уверенно сказать, что концы этих интервалов соответствуют крайним точкам объединённого множества решений, в чём можно легко убедиться.

Глава 6

Приложение

Рассмотрим практически важную задачу оценивания параметров электрохимического процесса формирования рыхлых осадков порошков металла. Постановка задачи вкратце такова: экспериментаторы имеют набор замеров, полученных в результате электрохимического процесса, и нуждаются в оценках его параметров, которые необходимы для решения вопроса об остановке процесса с высокой точностью по времени, основываясь на заранее полученных значениях.

Но получение оценок параметров процесса осложняется неточностью информации, полученной при измерении его характеристик, а также невозможностью применения традиционных вероятностных методов обработки погрешностей измерений. При небольшом количестве наблюдений стандартные методы теоретико-вероятностной математической статистики дают очень широкие доверительные интервалы оценок, так что применять их для отслеживания момента остановки процесса невозможно. Нам необходимо находить более точные двусторонние оценки состояния.

Для достижения поставленной цели можно использовать подходы интервального анализа, оперирующие с заданной оценкой максимального значения (по модулю) погрешностей. Они в самом деле позволяют получить искомые интервалы оценки состояний с более узкой шириной.

6.1. Постановка задачи

Для описания процесса формирования рыхлых осадков порошков металла была предложена следующая функция

$$W(t) = \frac{D + t}{R + t}, \tag{6.1}$$

которая была получена эмпирическим способом [21, 22]. D, R — параметры, измеряемые в секундах, на которые налагаются условия $R > D > 0$. Время (t, сек) — независимый аргумент.

Исходными данными для обработки являются шесть выборок измерений, по 19 измерений каждая:

$$\{t_{n,m}, W_{n,m}\}, \quad t_{n,m} \in [0, 2400], \quad n = 1, \ldots, 19, \ m = 1, \ldots, 6.$$

На этапе предобработки выполняем очистку измерений от выбросов и затем формируем одну большую выборку $\{t_n, W_n\}_{n=1}^{100}$, пользуясь тем, что все моменты времени, в которые сделаны измерения, различны. В дальнейшем будем обозначать полученную общую выборку базовых измеренных значений [20] как $\{(t_n, W_n)\}_{n=1}^{100}$. Далее осуществляем интервализацию этих базовых значений для зависимой переменной W, добавляя уравновешенный интервал, который соответствует абсолютной погрешности ε:

$$\boldsymbol{W}_n = W_n + [-\varepsilon, \varepsilon].$$

Моменты времени t_n, в которые производятся измерения, считаются заданными точно. Итак, получаем выборку интервальных данных $\{t_n, \boldsymbol{W}_n\}_{n=1}^{100}$.

Для решения задачи восстановления зависимости по интервальным данным мы подставляем данные из нашей выборки в равенство (6.1), получая в результате интервальную систему уравнений вида

$$\begin{cases} \dfrac{D + t_n}{R + t_n} = \boldsymbol{W}_n, \\ \quad n = 1, 2, \ldots, 100. \end{cases} \tag{6.2}$$

Далее нужно найти её множество решений и выбрать из него точку, в случае пустоты этого множества решений, найти точку, на которой достигается максимальная совместность этой системы. Отметим, что так как интервальность присутствует лишь в правой части этой системы, то её объединённое множество решений совпадает с допусковым множеством решений, и поэтому можно не различать их и говорить просто о «множестве решений».

Величина ε, вообще говоря, неизвестна точно, и в процессе обработки данных мы даже можем варьировать её. Её начальное значение было задано как $\varepsilon_{\text{init}} = 0.0175$. Но при этом множество решений системы (6.2) оказывается пустым. С помощью максимизации распознающего функционала множества решений системы (6.2) можно найти предельное значение для ε, при котором множество решений непусто, и оно равно $\varepsilon_{\text{crit}} = 0.0232$. Основываясь на этом результате, было принято решение задать для практического расчёта ограничение с 10%-ым запасом, т.е. $\varepsilon_{\text{w}} = 0.0255$.

6.2. Получение оценок параметров

В случае непустоты множества решений системы (6.2) часто бывают необходимы его оценки, в частности, внешняя оценка. Для решения этого вопроса можно воспользоваться тем обстоятельством, что система (6.2) может быть приведена к интервальной системе линейных алгебраический уравнений (ИСЛАУ), имеющей следующий вид:

$$\begin{cases} (1,\ -\boldsymbol{W}_n)\begin{pmatrix} D \\ R \end{pmatrix} = t_n \cdot (\boldsymbol{W}_n - 1), \\ \quad n = 1, 2, \ldots, 100. \end{cases} \tag{6.3}$$

Найдя известными методами внешнюю оценку для её множества решений, ответим на поставленный выше вопрос. Следует лишь иметь в виду, что в системе (6.3) интервальные параметры являются связанными, так как одновременно входят в матрицу и в правую часть. Как следствие, известные интервальные численные методы, ориентированные на решение ИСЛАУ без связей, с независимыми параметрами, могут давать в этой задаче довольно грубые оценки. По этой причине для получения более точных оценок лучше рассматривать задачу в её изначальной дробно-линейной форме (6.2).

В общем случае для отыскания оценки параметров можно использовать метод максимума совместности, в котором оценкой берётся точка максимума распознающего функционала множества решений системы (6.2) (см. [8, 23]). Так как независимые аргументы восстанавливаемой функции задаются точно, то неважно какой именно функционал будет использован, объединённого или допускового множеств решений: в данном случае они совпадают.

С помощью реализованных в библиотеке `IntvalPy` функций была выполнена глобальная оптимизация распознающего функционала Uni и найден его максимум, который оказался примерно равен 0.0018. Аргумент максимума функционала примерно равен $\arg\max \mathrm{Uni} \approx (170.69, 223.72) = (c_1, c_2)$. Интересно сравнить этот результат с оценкой параметров $(180.7, 235.5)$, которая была получена для той же задачи в работе [11] с помощью метода наименьших квадратов.

Отметим, что независимый аргумент t (время) дважды встречается в функции (6.1) для описания физического процесса. Выше мы рассмотрели решение задачи восстановления зависимости в случае, когда t является точной величиной. Если же t известно неточно и задаётся интервалами $\boldsymbol{t}_n$, то нужно озаботиться вычислением области значений отображения по $t \in \boldsymbol{t}_n$. В частности, естественное интервальное расширение здесь неприменимо из-за не единственности вхождения переменной. Для

преодоления этой проблемы можно видоизменить выражение и работать с его эквивалентной формой:

$$W(t) = \frac{D + t + R - R}{R + t} = 1 + \frac{D - R}{R + t}. \tag{6.4}$$

Опираясь на специфические свойства множества решений, изложим способ для вычисления внешней оценки множества решений для частного случая. Будет удобно заменить исходную задачу нахождения внешнего бруса для объединённого множества решений $\Xi_{uni}(F, t, \boldsymbol{W})$ на эквивалентную совокупность подзадач. Каждая из них требует нахождения отдельных покомпонентых оценок множества решений $\min\{x_\nu \mid x \in \Xi_{uni}(F, t, \boldsymbol{W})\}$ и $\max\{x_\nu \mid x \in \Xi_{uni}(F, t, \boldsymbol{W})\}$, где $\nu = 1, 2$. Далее для определённости рассмотрим нахождение максимума первой компоненты, остальные оценки находятся аналогично. Для нахождения интервала изменения параметра D мы используем идею, которая аналогична основной идее методов дробления графика (см. [24, 2]): оценки находятся на основе информации о сечениях множества решений прямыми, параллельными координатным осям.

Для нахождения интервала $\boldsymbol{D}$ изменения параметра D выразим его из равенства (6.1), а затем подставим в полученную формулу интервальную оценку $\boldsymbol{W}$ и значение параметра R, равное c_2 и вычисленное ранее:

$$\boldsymbol{D} = \boldsymbol{W} \cdot (c_2 + t) - t.$$

Далее, подставляя в это соотношение моменты времени t_n и интервалы $\boldsymbol{W}_n$, $n = 1, \ldots, 100$, мы получим систему равенств

$$\begin{cases} \boldsymbol{D} = \boldsymbol{W}_n \cdot (c_2 + t_n) - t_n, \\ \qquad n = 1, 2, \ldots, 100, \end{cases} \tag{6.5}$$

из которой пересечением отдельных интервалов $\boldsymbol{D}$ находится начальная оценка $\boldsymbol{d} = [d^{(0)}, d^{(1)}]$ среза множества решений, т. е. интервал в котором варьируется величина D при фиксированном значении параметра R.

Учитывая вид множества решений, которое было получено с помощью перебора на сетке в работе [11], нетрудно понять, что если интервал $\boldsymbol{d}$ вырождается в точку, то мы достигли наибольшего или наименьшего значения соответствующей компоненты точек из множества решений, т. е. получили точную верхнюю или точную нижнюю оценку этой компоненты.

Опишем более подробно алгоритм получения $D_{\sup}$, т. е. наименьшего значения первой компоненты точек из множества решений. Сначала найдём какое-нибудь значение f^0, при котором уравнение (6.5) не имеет

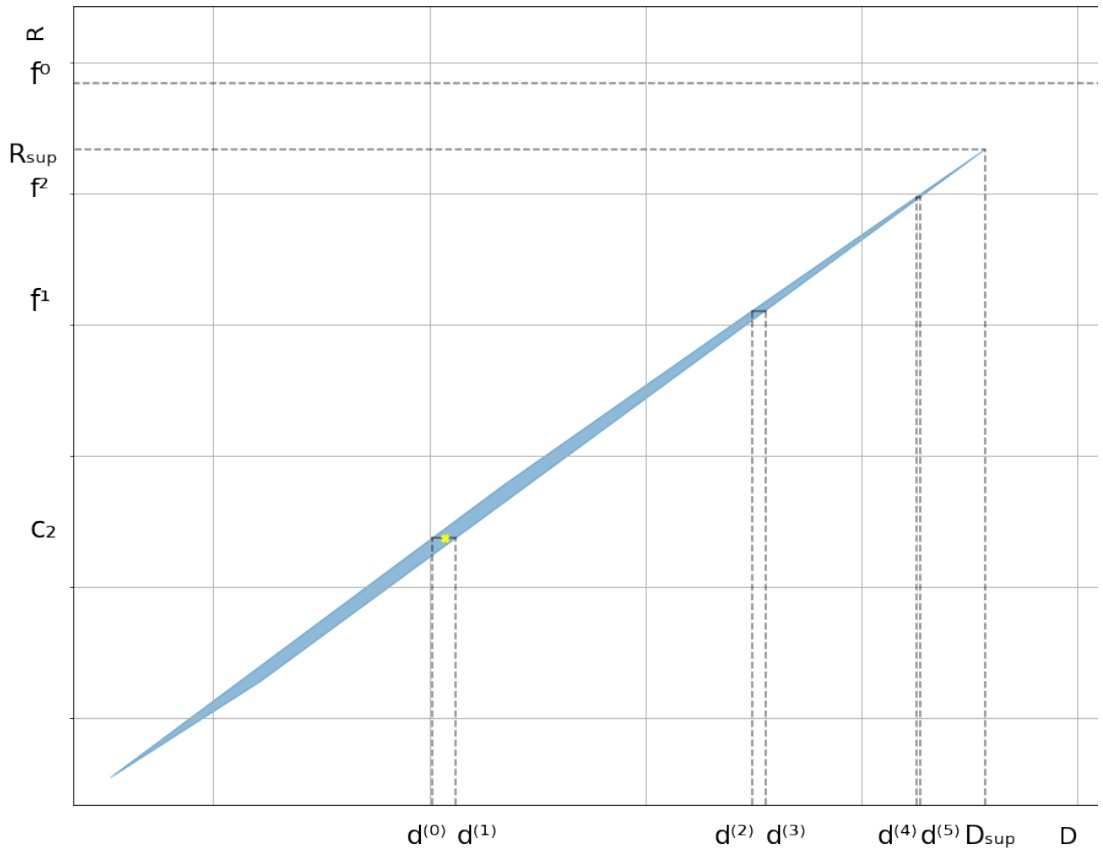


Рис. 6.1 – Получение внешней оценки первого аргумента.

решений. Проверить существование решений можно с помощью подстановки выбранного f^0 в систему (6.5) и пересечения отдельных $\boldsymbol{D}$. Если получаем непустой интервал, то система разрешима, иначе — неразрешима. После нахождения f^0, доставляющего неразрешимость системе, мы, фактически, получили интервал $\boldsymbol{r} = [c_2, f^0]$, где локализован максимум значений второй компоненты точек из множества решений. Организуя его дробление и проверку существования решений системы (6.5), мы сможем уточнить этот максимум. Более точно, станем последовательно дробить интервал $\boldsymbol{r}$ и его потомки пополам, подставляя их середины в уравнение (6.5) и проверяя пустоту/непустоту пересечений отдельных $\boldsymbol{D}$.

Если на каком-то шаге последовательного дробления мы получаем непустой интервал (система (6.5) неразрешима), то отбрасываем ту часть интервала, где значения меньше его центральной точки («нижнюю» часть), иначе — «верхнюю». После этого переходим на следующий шаг с новым интервалом $\boldsymbol{r}$. Описанные действия выполняются до тех пор пока ширина интервала $\boldsymbol{r}$ больше заданной величины, или же пока не исчерпано выделенное на алгоритм количество шагов.

В целом, опираясь на высокоуровневую функциональность интервальной библиотеки `IntvalPy`, удалось распознать разрешимость системы 6.2 и реализовать алгоритм для поиска внешней оценки множества решений практически значимой задачи. Отметим, что построенный алгоритм

является последовательно гарантирующим, что обеспечивает получение интервальной оценки параметров, даже если алгоритм будет прерван при исчерпании выделенного ему количества шагов (см. [2]).

Глава 7

Сравнение с другими библиотеками

Существует немало интервальных библиотек на самых разных языках программирования, среди которых C/C++, MATLAB/Octave, Julia, Java и др. Для демонстрации возможностей библиотеки `IntvalPy` был проведён качественный анализ её функциональности, а также реализованы вычислительные тесты для сравнения с другими модулями на языке Python.

7.1. Численные результаты

Целью вычислительных тестов является сравнение быстродействия кода по отношению к другим реализациям. Для проведения подобного рода экспериментов был реализован простейший алгоритм интервальной глобальной оптимизации, и его работа рассматривалась только на библиотеках, доступных на языке Python. Из немалого количества интервальных библиотек, написанных к настоящему моменту для языка Python, практический интерес имеют, как нам кажется, только две — `PyInterval` [19], `PyIbex`. Возможность использовать другую реализацию интервального типа на одном и том же языке позволяет корректно сравнить вычислительную эффективность.

В качестве тестовых задач глобальной оптимизации были рассмотрены целевые функции различной сложности, имеющие много локальных экстремумов, с различной трудоёмкостью решения. В первую очередь решались задачи безусловной оптимизации, в частности, использовались тесты из работы [18]. Таблица 7.1 демонстрирует полученные результаты.

Сравнивая результаты, полученные различными интервальными па-

Таблица 7.1 – Численные результаты безусловной глобальной оптимизации.

		PyInterval			PyIbex			IntvalPy			
F	Dim	N_{it}	ϵ	t_{cpu}	N_{it}	ϵ	t_{cpu}	N_{it}	ϵ	t_{cpu}	
										`double`	`ext.`
Branin (RC)	2	160	7.1^{-15}	0.301	164	8.88^{-15}	0.047	142	8.9^{-15}	0.107	0.517
Bohachevsky (B_m)	2	66	6.66^{-15}	0.149	66	6.66^{-15}	0.016	66	6.44^{-15}	0.061	0.278
Beale (BL)	2	342	5.66^{-15}	0.833	342	5.66^{-15}	0.087	342	5.66^{-15}	0.373	1.637
Booth (BO)	2	146	7.9^{-15}	0.214	146	7.9^{-15}	0.035	146	7.9^{-15}	0.079	0.316
De Joung (DJ)	3	80	8.73^{-15}	0.07	80	8.73^{-15}	0.011	80	8.73^{-15}	0.029	0.157
Easom (ES)	2	63	8.1^{-15}	0.116	63	8.55^{-15}	0.016	70	4.44^{-15}	0.061	0.217
Eggholder	2	10000	1.06^{-5}	246.7	10000	1.06^{-5}	5.45	10000	1.06^{-5}	15.94	57.32
Rastrigin (RT_n)	4	10000	4.26^{-14}	42.73	10000	4.26^{-14}	78.03	129	7.11^{-15}	0.311	1.137
Shaffner № 4	2	10000	2.83^{-7}	27.3	10000	2.83^{-7}	3.897	10000	2.83^{-7}	11.57	68.18
Levy № 13	2	10000	1.7^{-14}	30.57	10000	3.4^{-14}	4.085	68	8.57^{-15}	0.097	0.468

F — целевая функция, Dim — размерность задачи, N_{it} — ограниченное количество итераций для остановки оптимизации, ϵ — минимальная допустимая погрешность, t_{cpu} — усреднённое 50-ю запусками время выполнения задачи.

кетами, видим, что модуль `IntvalPy`, в целом, работает медленнее на интервальных операциях, чем другие библиотеки. Это связано с наличием дополнительных проверок типов при определении операций, при создании интервалов, а также использование типа данных повышенной точности вычислений. Например, в интервальном классе модуля `IntvalPy` всякая арифметическая операция проверяет к какой интервальной арифметике относится объект, что влияет на дальнейшее определение операций с ним. В будущем, возможно, стоит ожидать интеграции модуля `IntvalPy` с языком `C/C++` для дальнейшей модификации интервального класса, что должно в существенной степени увеличить скорость выполнения кода.

Помимо скорости решения задач оптимизации стоит обратить внимание на тот факт, что некоторые задачи так и не были до конца решены из-за исчерпания выделенного количества итераций. Однако, если такая ситуация возникала при выполнении в одной из библиотек, то не очевидно, что подобная ситуация возникнет и в другой реализации. По-видимому, большое различие в трудоёмкости решения одних и тех же задач различными библиотеками связано с тем, что некоторые из них используют направленные округления, результатами которых являются более широкие интервалы по сравнению с интервалами, полученными с повышенной точностью. В нелинейных задачах оптимизации, где рассматриваются сложные целевые функции, с большим количеством локальных экстремумов, это обстоятельство может потребовать бóльшего количества итераций для остановки алгоритма.

7.2. Качественный анализ

Целью качественного анализа 7.2 является сравнение возможностей, которые предоставляют другие интервальные библиотеки. Для этого рассмотрим подробнее их функциональность, точнее, такие её аспекты, как возможность работы с различными интервальными арифметиками, возможность выполнения матричных операций, наличие готовых инструментов для визуализации решений, наличие встроенных процедур для вычисления внешних и/или внутренних оценок параметров линейных систем, нахождение решений уравнений, а также возможность решения задач оптимизации.

Таблица 7.2 – Качественное сравнение интервальных библиотек.

Библиотека	Арифм.	Матричные операции	Визуал.	Оценка параметров	Решение уравнений	Оптим.
`IntvalPy`	+	+	+	+	+	+
`PyInterval`	+	−	−	−	+	−
`PyIbex`	−	+	+	−	+	+
`Octave` [1]	−	+	−	+	+	−
`WM` [2]	+	+	−	−	−	−
`JInterval`	+	+	+	+	+	+
`JI` [3]	−	+	+	−	+	+
`IntLab`	+	+	−	+	+	+

Стоит отметить, что инструменты для нахождения решений нелинейных систем уравнений в библиотеке `IntvalPy` хотя и присутствуют, но не очень развиты (как и в некоторых других библиотеках). Реализованные алгоритмы являются простейшими, в отличие от методов для линейных систем, среди которых присутствуют наиболее современные и развитые на данный момент численные методы.

7.3. Заключение

В работе изложены особенности разработанной библиотеки `IntvalPy`, главные идеи, положенные в её основу, а также её архитектура и реализация. Объяснены преимущества и достоинства библиотеки, в число которых входят востребованные сегодня кроссплатформенность, скорость вычислений и наиболее современные алгоритмы.

Библиотека позволяет учитывать специфические запросы пользователей, например, возможность отключения определённых опций в случае необходимости. Другой важной характеристикой созданного модуля является его практическая ориентированность — наличие в его составе готовых численных методов для решения различных прикладных задач интервального анализа. Кроме того, библиотека `IntvalPy` продолжает развиваться и в неё добавляется новый инструментарий.

[1] Подразумевается интервальная библиотека на свободно распространяемой программной системе `GNU Octave`.

[2] Интервальные возможности проприетарной системы `Wolfram Mathematica`.

[3] `JuliaIntervals`.

Литература

1. Sergei I. Zhilin — JInterval Library: Principles, Development, and Perspectives // Reliable Computing 19, 2014 pp. 229-247.

2. Шарый С.П. – Конечномерный интервальный анализ. Новосибирск: 2011. Электронная книга, доступная на http://www.nsc.ru/interval/Library/InteBooks

3. Shary S.P. Interval regularization for imprecise linear algebraic equations. – Новосибирск, 2018. – 21 с. – Депонировано в электронном репозитории arXiv.org, регистрационный номер arXiv:1810.01481.

4. Oettli W., Prager W. Compatibility of approximate solution of linear equations with given error bounds for coefficients and right-hand sides // Numerische Mathematik. 1964. V. 6. P. 405–409.

5. Kearfott R.B., Nakao M., Neumaier A., Rump S., et al. Standardized notation in interval analysis // Вычисл. технологии. 2010. Т. 15. №1. С. 7–13.

6. С.П.Шарый, И.А.Шарая - Распознавание разрешимости интервальных уравнений и его приложения к анализу данных. // Вычислительные технологии, Том 18, No 3, 2013, стр. 80-109.

7. Демиденко Е.З. Комментарий II к статье А.П. Вощинина, А.Ф. Бочкова и Г.Р. Сотирова «Метод анализа данных при интервальной нестатистической ошибке» // Заводская лаборатория. 1990.Т. 56. No7. С. 83–84.

8. С.П.Шарый – Разрешимость интервальных линейных уравнений и анализ данных с неопределённостями // Автоматика и Телемеханика, No 2, 2012

9. Ostanina, T. N. Modelling the dynamic growth of copper and zinc dendritic deposits under the galvanostatic electrolysis conditions / T.N. Ostanina, V. M. Rudoi, A. V. Patrushev, A. B. Darintseva, A. S. Farlenkov // J. Electroanal. Chem. – 2015. – Vol. 750. – P. 9-18.

10. Ostanina, T. N. Determination of the surface of dendritic electrolytic zinc powders and evaluation of its fractal dimension / T.N. Ostanina, V.M. Rudoy, V.S. Nikitin, A.B. Darintseva, O.L. Zalesova, N.M.Porotnikova // Russ. J. Non-Ferr. Met. – 2016. – Vol. 57 – P. 47–51. DOI: 10.3103/S1067821216010120.

11. Sergey I. Kumkov, Vyacheslav S. Nikitin, Tatyana N. Ostanina, and Valentin M. Rudoy - Interval processing of electrochemical data. // Journal of Computational and Applied Mathematics

12. Irene A. Sharaya. Boundary intervals and visualization of AE-solution sets for interval systems of linear equations. In Book of abstracts of 15th GAMM-IMACS International Symposium on Scientic Computing, Computer Arithmetics and Veri

    ed Numerics SCAN-2012,Novosibirsk, Russia, September 23-29, 2012.

13. И.А. Шарая - Метод граничных интервалов для визуализации полиэдральных множеств решений // Вычислительные технологии, Том 20, No 1, 2015, стр. 75-103.

14. А.Н. Баженов - Интервальный анализ. Основы теории и учебные примеры. Санкт-Петербург: 2020

15. Стецюк П.И. Субградиентные методы ralgb5 и ralgb4 для минимизации овражных выпуклых функций // Вычислительные технологии Том 22, No 2, 2017, стр. 127 - 149.

16. Shary S.P. On optimal solution of interval linear equations // SIAM Journal on Numerical Analysis. – 1995. – Vol. 32, No. 2. – P. 68–630.

17. J. Rohn Enclosing Solutions of Overdetermined Systems of Linear Interval Equations // Reliable Computing 2 (2) (1996), pp. 167-171.

18. Abdel-Rahman Hedar, A. Ahmed «Studies on metaheuristics for continuous global optimization problems» // Kyoto University, June 2004

19. Stefano Taschini Interval Arithmetic: Python Implementation and Applications // Proc. SciPy 2008, G. Varoquaux, T. Vaught, J. Millman (Eds), pp. 16–22

20. Баженов А.Н., Жилин С.И., Кумков С.И., Шарый С.П., Обработка и анализ данных с интервальной неопределённостью. – Новосибирск, 2022. Адрес доступа: `http://www.nsc.ru/interval/Library/ApplBooks/InteDataProcessing.pdf`

21. Ostanina, T. N. Modelling the dynamic growth of copper and zinc dendritic deposits under the galvanostatic electrolysis conditions / T.N. Ostanina, V. M. Rudoi, A. V. Patrushev, A. B. Darintseva, A. S. Farlenkov // J. Electroanal. Chem. – 2015. – Vol. 750. – P. 9-18.

22. Ostanina, T. N. Determination of the surface of dendritic electrolytic zinc powders and evaluation of its fractal dimension / T.N. Ostanina, V.M. Rudoy, V.S. Nikitin, A.B. Darintseva, O.L. Zalesova, N.M.Porotnikova // Russ. J. Non-Ferr. Met. – 2016. – Vol. 57 – P. 47–51. DOI: 10.3103/S1067821216010120.

23. С.П.Шарый – Сильная согласованность в задаче восстановления зависимостей при интервальной неопределённости данных // Вычислительные технологии Том 22, No 2, 2017

24. Sergey P. Shary, Graph subdivision methods in interval global optimization // Constraint Programming and Decision Making. Martine Ceberio and Vladik Kreinovich, editors. – Heidelberg-New York: Springer, 2014. – P. 153–170. (Studies in Computational Intelligence; vol. 539).