Научная статья

УДК 519.68
DOI 10.25205/1818-7900-2022-20-4-5-23

IntvalPy – библиотека интервальных вычислений на языке Python

Артём Станиславович Андросов, Сергей Петрович Шарый

Федеральный исследовательский центр информационных и вычислительных технологий
Новосибирск, Россия

artem.androsov@gmail.com, https://orcid.org/0009-0008-9043-0157

shary@ict.nsc.ru, https://orcid.org/0000-0002-9454-7626

Аннотация

В статье рассматривается библиотека IntvalPy, реализующая интервальные вычисления на языке Python. В отличие от других существующих интервальных библиотек IntvalPy предоставляет возможность работы как с классической интервальной арифметикой, так и с полной интервальной арифметикой Каухера. Кроме того, библиотека была разработана с учетом стандарта IEEE 1788–2015 на интервальные вычисления на ЭВМ, что гарантирует высокую точность результатов, а также совместимость с другими программными продуктами. Верхнеуровневая функциональность библиотеки IntvalPy реализует новейшие методы для распознавания и оценивания множеств решений интервальных линейных систем уравнений, вычисления их формальных решений, а также визуализации множеств решений интервальных уравнений и систем уравнений. В качестве примера приложения библиотеки была решена практически важная задача оценивания параметров электрохимического процесса формирования рыхлых осадков порошков металла. Кроме того, были проведены численные и качественные сравнения с другими интервальными библиотеками для демонстрации функциональных возможностей и оптимальности реализованных интервальных классов.

Ключевые слова

интервальный анализ, библиотека интервальных вычислений, Python, неточные данные, интервальные системы уравнений, визуализация

Для цитирования

Андросов А. С., Шарый С. П. IntvalPy – библиотека интервальных вычислений на языке Python // Вестник НГУ. Серия: Информационные технологии. 2022. Т. 20, № 4. С. 5–23. DOI 10.25205/1818-7900-2022-20-4-5-23

IntvalPy — a Python Interval Computation Library

Artem S. Androsov, Sergey P. Shary

Federal Research Center for Information and Computational Technologies
Novosibirsk, Russian Federation

artem.androsov@gmail.com, https://orcid.org/0009-0008-9043-0157

shary@ict.nsc.ru, https://orcid.org/0000-0002-9454-7626

Abstract

The article presents the IntvalPy library which implements interval computations in Python. Unlike other existing interval libraries, IntvalPy allows one to work with both classical interval arithmetic and complete Kaucher interval arithmetic. In addition, the library was developed taking into account the IEEE 1788-2015 standard for interval arithmetic on digital computers, which guarantees high accuracy of the results and compatibility with the other existing software

products. The top-level functionality of the IntvalPy library implements state-of-the-art methods for recognizing and estimating solution sets for interval linear systems of equations, computing their formal solutions, and visualizing solution sets for interval equations and systems of equations. As an example of the library application, we solve the practically important problem of estimating the parameters of the electrochemical process of the formation of loose metal powder precipitates. Additionally, numerical computation was carried out, as well as qualitative comparisons with other interval libraries, in order to demonstrate the functionality and optimality of implemented interval classes.

Keywords

interval analysis, interval computing library, Python, inaccurate data, interval systems of equations, visualization

For citation

Androsov A. S., Shary S. P. IntvalPy — a Python Interval Computation Library. V*estnik NSU. Series: Information Technologies*, 2022, vol. 20, no. 4, pp. 5–23. (in Russ.) DOI 10.25205/1818-7900-2022-20-4-5-23

Введение

Настоящая работа посвящена созданию библиотеки программ на языке Python, реализующей интервальные вычисления и численные методы интервального анализа для решения ряда стандартных математических задач. Это современные методы для определения существования решений как квадратных, так и переопределенных интервальных систем линейных алгебраических уравнений, вычисление внешних и внутренних оценок множеств решений, а также визуализации множеств решений интервальных уравнений и систем уравнений.

Перед нами стояла задача реализовать на популярном алгоритмическом языке развитый инструментарий современного интервального анализа, главным образом касающийся интервальных систем линейных алгебраических уравнений и родственных к ним постановок.

Мотивацией для данной работы послужил также 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 ее разработчиками.

Кроме того, среди большого количества программных модулей, реализованных к настоящему моменту для Python, отсутствуют высокоуровневые интервальные средства, необходимые для решения таких практически важных задач, как распознавание разрешимости интервальных систем уравнений, оценивание и визуализация их множеств решений и т. п. Кроме того, нет хороших продуктов для визуализации множеств решений линейных неравенств $A\,x \leq b$ для случаев двух и трех переменных. В силу известных результатов такая функциональность позволила бы реализовать визуализацию множества решений для интервальных линейных уравнений. Другой ключевой мотивацией нашей работы было то, что в большинстве созданных интервальных библиотек для Python отсутствует возможность работать в интервальных арифметиках, отличных от классической. Но классическая интервальная арифметика не имеет обратных элементов относительно основных арифметических операций и у нее плохие порядковые свойства. Таким образом, порой необходимо работать в полной интервальной арифметике или арифметике Каухера. Все вышеперечисленные соображения послужили причиной для создания указанной библиотеки IntvalPy (см. [1]).

1. Теория

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

$$[a, b] \coloneqq \{x \in R \mid a \leq x \leq b\}.$$

Система обозначений в работе соответствует неформальному международному стандарту в интервальном анализе [2]. В частности, интервалы и интервальные величины обозначаются

жирным шрифтом, к примеру, ***a, b, c, ...***, тогда как точечные величины 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. $\boldsymbol{a} = \left[\underline{\boldsymbol{a}}, \overline{\boldsymbol{a}}\right]$. Интервал называется *вырожденным* при $\underline{\boldsymbol{a}} = \overline{\boldsymbol{a}}$ и *невырожденным* иначе.

Прежде всего определим наиболее важные и часто встречающиеся арифметические операции между интервалами. Основным руководящим принципом будет служить соотношение

$$\boldsymbol{a} * \boldsymbol{b} = \{a * b \mid a \in \boldsymbol{a},\ b \in \boldsymbol{b}\}, \quad * \in \{+,\ -,\ \cdot,\ /\},$$

согласно которому интервальные арифметические операции выполняются «по представителям». Конструктивные определения интервальных арифметических операций имеют вид:

сложение: $\boldsymbol{a} + \boldsymbol{b} = \left[\underline{\boldsymbol{a}} + \underline{\boldsymbol{b}}, \overline{\boldsymbol{a}} + \overline{\boldsymbol{b}}\right]$,

вычитание: $\boldsymbol{a} - \boldsymbol{b} = \left[\underline{\boldsymbol{a}} - \overline{\boldsymbol{b}}, \overline{\boldsymbol{a}} - \underline{\boldsymbol{b}}\right]$,

умножение: $\boldsymbol{a} \cdot \boldsymbol{b} = [\min S, \max S]$, где $S = \underline{\boldsymbol{a}}\,\underline{\boldsymbol{b}}, \underline{\boldsymbol{a}}\,\overline{\boldsymbol{b}}, \overline{\boldsymbol{a}}\,\underline{\boldsymbol{b}}, \overline{\boldsymbol{a}}\,\overline{\boldsymbol{b}}$,

деление: $\boldsymbol{a}/\boldsymbol{b} = \boldsymbol{a} \cdot \left[\mathbf{1}/\overline{\boldsymbol{b}}, 1/\underline{\boldsymbol{b}}\right]$, если $0 \notin b$.

Алгебраическая система, носителем которой является множество всех интервалов вещественной оси $\mathbb{R}$ с определенными для них интервальными арифметическими операциями сложения, вычитания, умножения и деления, называется *классической интервальной арифметикой* и обозначается $\mathbb{IR}$.

К сожалению, алгебраические свойства этой арифметики довольно скудны по сравнению с привычными числовыми системами, такими 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. Неудовлетворительны порядковые свойства относительно упорядочения по включению.

Из-за отсутствия дистрибутивности невозможно приводить подобные члены, что является серьезным недостатком интервальной арифметики. Для преодоления этой проблемы необходимо в значительной степени изменить данную арифметику как алгебраическую систему, что ставит под вопрос саму 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 множеству обычных (правильных) интервалов $\mathbb{IR} = \{[a, b] \mid a, b \in \mathbb{R}, a \leq b$ и вещественных чисел. Для перехода от неправильных к правильным интервалам применяется операция *дуализации* $\mathrm{dual}: \mathbb{KR} \to \mathbb{KR}$.

Распространение операций сложения и вычитания на полную интервальную арифметику выполняется достаточно просто, а явные формулы для умножения удобно выделить в следующие подмножества, которые можно представить в виде табл. 1:

$\mathcal{P} := \{\boldsymbol{a} \in \mathbb{KR} \mid \underline{\boldsymbol{a}} \geq 0, \overline{\boldsymbol{a}} \geq 0\}$ – неотрицательные интервалы,

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

$-\mathcal{P} := \{\boldsymbol{a} \in \mathbb{KR} \mid -\boldsymbol{a} \in \mathcal{P}\}$ – неположительные интервалы,

$\mathrm{dual}\,\mathcal{Z} := \{\boldsymbol{a} \in \mathbb{KR} \mid \mathrm{dual}\,\boldsymbol{a} \in \mathcal{Z}\}$ – интервалы, содержащиеся в нуле.

Таблица 1

Умножение в арифметике Каухера

Table 1

Multiplication in Kaucher Arithmetic

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

2. Постулаты

При создании любого программного модуля разработчики руководствуются некоторыми основными требованиями. В данном разделе в порядке убывания значимости 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 четному 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]

5. Библиотека должна обеспечивать возможность работы в разных интервальных арифметиках (классическая, Каухера, Кэхэна 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.).

3. Архитектура библиотеки

Библиотека IntvalPy, написанная на языке программирования Python, реализует алгебраически замкнутую систему для работы с интервалами, для решения интервальных систем

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

как линейных, так и нелинейных уравнений, а также для визуализации множества решений интервальных линейных систем уравнений.

В этом разделе представлены основные идеи, которые определяют архитектуру библиотеки и ее 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 значительно 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 остается стандартным – округление к ближайшему числу с четной 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 распространенный модуль NumPy. Его применение особенно полезно потому, что позволяет создавать не только векторы, 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 нем реализуются циклы.

Реализация библиотеки IntvalPy соответствует схеме, показанной на рис. 1, где стрелки обозначают включение (подчинение) отдельных структурных элементов в другие элементы. Наиболее подробно разработаны низкоуровневые операции, а также численные методы для интервальных линейных систем.

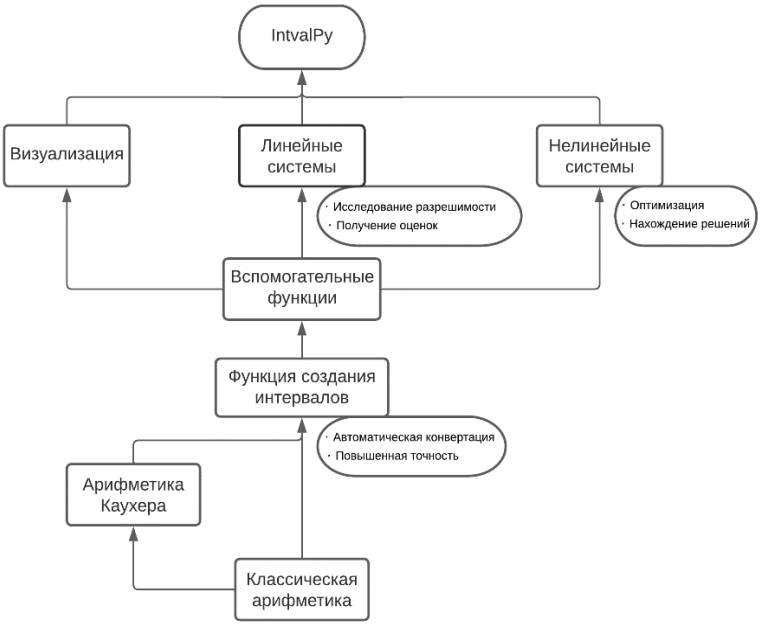

Рис. 1. Архитектура IntvalPy
Fig. 1. IntvalPy architecture

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 произвольной 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, которые были описаны ранее. В целом библиотека спроектирована как расширяемая, и все ее 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.

4. Функциональность

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

4.1. Низкоуровневая функциональность

Операции нижнего функционального слоя модуля IntvalPy, доступные для использования, – это операции интервальных арифметик, классической интервальной арифметики и полной арифметики 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, элементарные функции, а также различные характеристики интервалов, которые соответствуют стандарту IEEE 1788-2015. В табл. 2 резюмированы операции доступные для применения.

Таблица 2

Доступные операции в IntvalPy

Table 2

Available Operations in IntvalPy

Операции	**Описание**
+, -, ·, /	Арифметические операции
pow, abs, exp, log, sin, cos	Элементарные функции
matmul, dot, transpose	Матричные операции
$<, \leq, =, \geq, >, /=, \subset, \subseteq, \cap$	Бинарные отношения и операции
rad, wid, mid, mig, mag	Характеристики интервалов
dual, pro, opp, inv	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.

Отметим следующее. Поскольку программный модуль предоставляет возможность работы с интервальными векторами и матрицами, то не следует забывать о том, что аргумент передается 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.

```
import intvalpy as ip
import numpy as np

# Barth-Nuding system
A = ip.Interval([
    [[2, 4], [-2, 1]],
    [[-1, 2], [2, 4]]
])
b = ip.Interval([[-2, 2], [-2, 2]])
x = np.array([-0.2, 0.4])

ip.subset(A @ x, b)     # True
# A @ x = Interval(['[-1.6, 0]', '[0.4, 1.8]'])

a = A[0, 1].opp         # Interval(['[2, -1]'])
type(A[0, 1].opp)
# intvalpy.RealInterval.KaucherArithmetic
```

Листинг 1. Основные операции
Listing 1. Basic operations

Рассмотрим программный код в листинге 2, который реализует вычисление выражения из известного примера Румпа [3; 4]:

$$f(a,b) = 333.75 \cdot b^6 + a^2 \cdot (11 \cdot a^2 \cdot b^2 - b^6 - 121 \cdot b^4 - 2) + 5.5 \cdot b^8 + a/(2 \cdot b). \quad (1)$$

Этот пример иллюстрирует тот факт, что результаты численных расчетов на компьютере не обязательно должны приближаться к истинному ответу, несмотря на применение возрастающей точности вычислений (float, double, long double и т. д.).

```
import intvalpy as ip

ip.precision.extendedPrecisionQ = True
ip.precision.dps(40)

f = lambda a, b: 333.75 * b**6 + \
    a**2 * (11 * a**2 * b**2 - b**6 - \
    121 * b**4 - 2) + 5.5 * b**8 + a/(2*b)

a = ip.Interval(77617, 77617)
b = ip.Interval(33096, 33096)
f(a, b)
# Interval(['[-0.827396, -0.827396]'])
```

Листинг 2. Пример Румпа
Listing 2. Rump example

Интервальные библиотеки, реализующие интервальные вычисления с внешним направленным округлением, для примера Румпа дают довольно широкие интервалы (шириной порядка 10^{21}), которые содержат правильный ответ. К сожалению, эти ответы малопригодны для практического использования из-за своей огромной ширины. Если же увеличивать точность представления концов, то результирующий интервал будет сужаться. С помощью библиотеки IntvalPy, используя повышенную точность представления концов интервалов с 40 знаками после запятой, можно получить довольно узкие интервальные оценки результата.

Кроме явно указанных функций, в библиотеке IntvalPy есть и другие, например, zeros, eye, Shary, Neumeier и т. п. Использование этих функций помогает писать более короткие и выразительные программы, а также создавать тестовые задачи с хорошо изученными свойствами. Читатель может найти описания этих функций в документации, однако библиотека регулярно расширяется, и документация может несколько отставать от фактического состояния библиотеки. Впрочем, самую актуальную информацию можно найти непосредственно в исходном коде библиотеки, который расположен в программном репозитории GitHub.

4.2. Верхнеуровневая функциональность

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 переопределенных систем интервальных линейных уравнений. Для двух- и трехмерных интервальных линейных систем на основе метода граничных интервалов, предложенного И. А. Шарой [5, 6], реализованы функции для визуализации различных множеств решений. Кроме того, в библиотеке представлены функции вычисления и максимизации распознающих функционалов множеств решений, которые являются основой метода максимума согласования [7, 8] для восстановления зависимостей по интервальным данным.

Далее мы продемонстрируем, как именно применять IntvalPy на нескольких конкретных примерах.

В качестве первого примера разберем исследование разрешимости интервальной системы линейных алгебраических уравнений (ИСЛАУ), понимаемое, как непустота соответствующего множества решений (ее также называют «сильной разрешимостью», см. [9]). Ниже мы рассмотрим допусковое множество решений ИСЛАУ (см., к примеру, [8, 10]), и для его исследования применим *метод распознающего функционала*. Он сводит исходную задачу к безусловной максимизации некоторой специальной функции, называемой распознающим функционалом множества решений, и по итогам нахождения ее максимума делается заключение о пустоте/непустоте множества решений, а также получается точка, обладающая наибольшей совместимостью с данной системой уравнений.

Во фрагменте кода, приведенном на листинге 3, рассмотрена тестовая система, предложенная А. Ноймайером в работе [11].

```
1 import intvalpy as ip
2
3 A, b = ip.Neumeier(3, 3.33, infb=-1, supb=2)
4 x = np.array([-3, 7, 2])
5 print('Tol: ', ip.linear.Tol(A, b, x=x))
6 print('max Tol: ', ip.linear.Tol(A, b, maxQ=True))
```

Листинг 3. Метод распознающих функционалов
Listing 3. The method of recognizing functionals

Представленный код проверяет, принадлежит ли точка x допусковому множеству решений (в первом операторе print), а далее, с помощью максимизации распознающего функционала Tol, находит точку, которая в наибольшей степени совместна с рассматриваемой ИСЛАУ (во втором операторе print). Напомним, что для интервальных линейных систем уравнений распознающий функционал Tol обладает хорошими свойствами: он является непрерывным, вогнутым и достигает конечного максимума, что упрощает задачу нахождения его безусловного максимума. В частности, в функции ip.linear.Tol используется алгоритм негладкой оптимизации ralgb5, разработанный П.И. Стецюком [12].

Отметим, что помимо сильной разрешимости (относительно допускового множества решений) часто рассматривается также слабая разрешимость ИСЛАУ, понимаемая как непустота ее объединенного множества решений. Это множество решений является, по-видимому, наиболее популярным из различных множеств решений для интервальных систем уравнений. Его пустота/непустота тоже может быть исследована с помощью распознающего функционала Uni, но в общем случае эта задача является труднорешаемой (NP-трудной), а распознающий функционал Uni в общем случае является многоэкстремальным. При нахождении его глобального максимума большую роль играет выбор начального приближения, и это также характерно для максимизации распознающих функционалов нелинейных интервальных систем уравнений.

Полное исчерпывающее описание множеств решений интервальных систем уравнений невозможно и практически не нужно, так что в случае непустоты множеств решений обычно находят для них приближенные оценки в том или ином смысле, требуемом смыслом рассматриваемой задачи [10]. 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. Среди всех этих численных методов наиболее интересны для нас PSS-методы, которые позволяют находить оптимальную внешнюю интервальную оценку множеств решений и, кроме того, обладают свойством последовательной гарантии.

Напомним, что задача вычисления оптимальных или гарантированно близких к оптимальным оценок объединенного множества решений ИСЛАУ является NP-трудной, поэтому в библиотеке IntvalPy реализована гибридная версия PSS-методов, удобная для решения практических задач значительной размерности. Кроме того, эти методы позволяют решать переопределенные линейные интервальные системы, возникающие при обработке данных. В листинге 4 представлено численное решение модельной тестовой системы С.П. Шарого [13] с размерностью 20 × 20.

```
1  import intvalpy as ip
2
3  A, b = ip.Shary(20)
4  pss = ip.linear.PSS(A, b)
```

Листинг 4. Метод дробления решений
Listing 4. Solution splitting methods

В качестве последнего примера рассмотрим задачу визуализации объединенного множества решений ИСЛАУ (в библиотеке IntvalPy реализована также процедура визуализации допускового множества решений). Типичные подходы к визуализации полиэдральных множеств, какими являются множества решений ИСЛАУ, основаны на перечислении их вершин, но этот путь и основанные на нем процедуры имеют ряд недостатков. Они работают только с квадратными системами, плохо обрабатывают неограниченные и тощие множества решений (с пустой

внутренностью). В библиотеке IntvalPy визуализация множеств решений использует метод граничных интервалов, предложенный И.А. Шарой для исследования и отрисовки полиэдральных множеств [5, 6, 14]. Главным преимуществом этого подхода является возможность работать с неограниченными и тощими множествами решений, а также с линейными системами, в которых количество уравнений не равно количеству неизвестных.

```
1  import intvalpy as ip
2  %matplotlib notebook
3
4  A = ip.Interval([
5      [[-1, 1], [-2, 2], [-2, 2]],
6      [[-2, 2], [-1, 1], [-2, 2]],
7      [[-2, 2], [-2, 2], [-1, 1]]
8  ])
9  b = ip.Interval([[2, 2], [2, 2], [2, 2]])
10
11 bounds = [[-2, -2, -2], [2, 2, 2]]
12 ip.IntLinIncR3(A, b, alpha=0.5, s=0, \
13                 bounds=bounds, size=(11,11))
```

Листинг 5. Метод граничных интервалов
Listing 5. Boundary intervals method

Результат работы программы листинга 5 представлен слева на рис. 2, а справа, на этом же рисунке, – пример визуализации тощего множества решений переопределенной интервальной системы (2) линейных алгебраических уравнений.

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

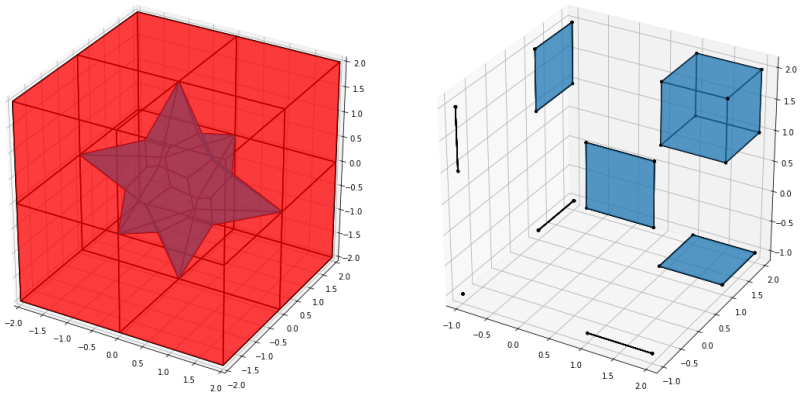

Рис. 2. Объединенные множества решений интервальных линейных систем
Fig. 2. United sets of solutions of interval linear systems

Поясним, что множество решений слева на рис. 2. является неограниченным. По этой причине оно было обрезано, и для понимания, где именно это произошло, плоскости обрезки окрашены в красный цвет.

5. 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 (см. [16]). 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.

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

$$W(t) = \frac{D + t}{R + t}, \tag{3}$$

которая была получена эмпирическим способом [15, 16]. D, R – параметры, измеряемые в секундах, на которые налагаются условия $R > D$. 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 [17] 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 $\{t_n, W_n\}_{n=1}^{100}$.

Для решения задачи восстановления зависимости по интервальным данным мы подставляем данные из нашей выборки в равенство (3), получая в результате интервальную систему уравнений вида

$$\begin{cases} \dfrac{D + t_n}{R + t_n} = \boldsymbol{W}_n, \\ n = 1{,}2, \ldots, 100. \end{cases} \tag{4}$$

Далее нужно найти ее 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 ее объединенное 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 ее. Ее начальное значение было задано как $\varepsilon_{\text{init}} = 0.0175$. Но при этом множество решений системы (4) оказывается пустым. С помощью максимизации распознающего функционала множества решений системы (4) 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 расчета ограничение с 10%-м запасом, т. е. $\varepsilon = 0.0255$.

В случае непустоты множества решений системы (4) часто бывают необходимы его оценки, в частности, внешняя оценка. Для решения этого вопроса можно воспользоваться тем обстоятельством, что система (4) может быть приведена к интервальной системе линейных алгебраический уравнений, имеющей следующий вид:

$$\begin{cases} (1, -\boldsymbol{W}_n)\begin{pmatrix} D \\ R \end{pmatrix} = t_n \cdot (\boldsymbol{W}_n - 1), \\ n = 1,2,\ldots,100. \end{cases} \tag{5}$$

Найдя известными методами внешнюю оценку для ее множества решений, ответим на поставленный выше вопрос. Следует лишь иметь в виду, что в системе (5) 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 ее изначальной дробно-линейной форме (4).

В общем случае для отыскания оценки параметров можно использовать метод максимума совместности, в котором оценкой берется точка максимума распознающего функционала множества решений системы (4) (см. [7, 8]). Так как независимые аргументы восстанавливаемой функции задаются точно, то неважно какой именно функционал будет использован, объединенного 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 $\arg\max \text{Uni} \approx \approx (170.69,223.72) = (c_1, c_2)$. Интересно сравнить этот результат с оценкой параметров (180.7,235.5), которая была получена для той же задачи в работе [18] с помощью метода наименьших квадратов.

Отметим, что независимый аргумент *t* (время) дважды встречается в функции (3) для описания физического процесса. Выше мы рассмотрели решение задачи восстановления зависимости в случае, когда *t* является точной величиной. Если же *t* известно неточно и задается 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 неединственности вхождения переменной. Для преодоления этой проблемы можно видоизменить выражение и работать с его эквивалентной формой:

$$W(t) = \frac{D + t + R - R}{R + t} = 1 + \frac{D - R}{R + t}. \tag{6}$$

Опираясь на специфические свойства множества решений, изложим способ вычисления внешней оценки множества решений для частного случая. Заменим исходную задачу нахож-

дения внешнего бруса для объединенного множества решений $\Xi_{uni}(F,t,\boldsymbol{W})$ на эквивалентную совокупность подзадач. Каждая из них требует нахождения отдельных покомпонентных 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 определенности 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. [10, 19]): оценки находятся на основе информации о сечениях множества решений прямыми, параллельными координатным осям.

Для нахождения интервала $\boldsymbol{D}$ изменения параметра $\boldsymbol{D}$ выразим его из равенства (3), 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 $\boldsymbol{W}_n$, где $n = 1, ..., 100$, мы получим систему равенств

$$\begin{cases} \boldsymbol{D} = \boldsymbol{W}_n \cdot (c_2 + t_n) - t_n, \\ n = 1,2,\ldots,100. \end{cases} \tag{7}$$

Из нее 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 [18], 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 (рис. 3).

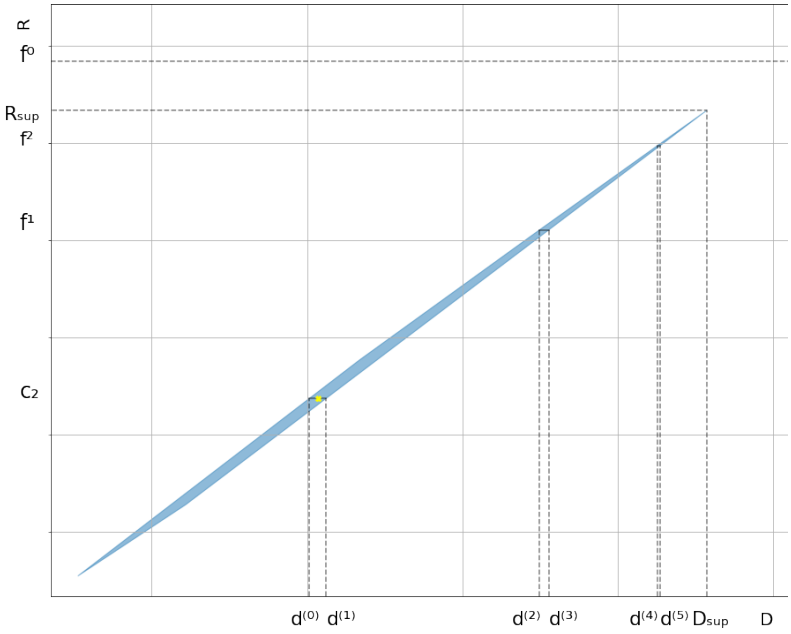

Рис. 3. Иллюстрация получения внешней оценки для значений первого измерения
Fig. 3. Illustration of obtaining an external estimate for the first measurements

Опишем более подробно алгоритм получения D_{sup}, т. е. наименьшего значения первой компоненты точек из множества решений. Сначала найдем какое-нибудь значение f^0, при котором уравнение (7) не имеет решений. Проверить существование решений можно с помощью под-

становки выбранного f^0 в систему (7) 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 (7), мы сможем уточнить этот максимум. Более точно, станем последовательно дробить интервал $\boldsymbol{r}$ и его потомки пополам, подставляя их середины в уравнение (7) и проверяя пустоту/непустоту пересечений отдельных $\boldsymbol{D}$.

Если на каком-то шаге последовательного дробления мы получаем непустой интервал (система (7) 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 (4) 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. [10]).

6. 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 созданной библиотеки IntvalPy был проведен качественный анализ ее функциональности, а также реализованы вычислительные тесты для сравнения с другими модулями на языке Python.

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 [20], 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 трудоемкостью решения. В первую очередь решались задачи безусловной оптимизации, в частности, использовались тесты из работы [21]. Табл. 3 демонстрирует полученные результаты.

Сравнивая результаты, полученные различными интервальными пакетами, 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,

Таблица 3

Численные результаты безусловной глобальной оптимизации

Table 3

Numerical Results of Unconditional Global Optimization

IntvalPy	t_{cpu}	ext.	0,517	0,278	1,637	0,316	0,157	0,217	57,32	1,137	68,18	0,468
		double	0,107	0,061	0,373	0,079	0,029	0,061	15,94	0,311	11,57	0,097
	∈		$8{,}90^{-15}$	$6{,}55^{-15}$	$5{,}66^{-15}$	$7{,}90^{-15}$	$8{,}73^{-15}$	$4{,}44^{-15}$	$1{,}06^{-15}$	$7{,}11^{-15}$	$2{,}83^{-15}$	$8{,}57^{-15}$
	N_{it}		142	66	342	146	80	70	10000	129	10000	68
PyIbex	t_{cpu}		0,047	0,016	0,087	0,035	0,011	0,016	5,450	78,03	3,897	4,085
	∈		$8{,}88^{-15}$	$6{,}66^{-15}$	$5{,}66^{-15}$	$7{,}90^{-15}$	$8{,}73^{-15}$	$5{,}55^{-15}$	$1{,}06^{-15}$	$4{,}26^{-14}$	$2{,}83^{-7}$	$3{,}14^{-14}$
	N_{it}		164	66	342	146	80	63	10000	10000	10000	10000
PyInterval	t_{cpu}		0,301	0,149	0,833	0,214	0,070	0,116	246,7	42,73	27,30	30,57
	∈		$7{,}10^{-15}$	$6{,}66^{-15}$	$5{,}66^{-15}$	$7{,}90^{-15}$	$8{,}73^{-15}$	$8{,}10^{-15}$	$1{,}06^{-15}$	$4{,}26^{-14}$	$2{,}83^{-7}$	$1{,}70^{-14}$
	N_{it}		160	66	342	146	80	63	10000	10000	10000	10000
	dim		2	2	2	2	3	2	2	4	2	2
	F		Branin (*RC*)	Bohachevsky (B_m)	Beale (*BL*)	Booth (*BO*)	De Joung (*DJ*)	Easom (*ES*)	Eggholder	Rastrigin (RT_n)	Schaffer № 4	Levy № 13

Примечание. F – целевая функция, dim – размерность задачи, N_{it} – ограниченное количество итераций для остановки оптимизации, ∈ – минимальная допустимая погрешность, t_{cpu} – усредненное 50-ю запусками время выполнения задачи, double – двойная точность, extended – повышенная точность.

то не очевидно, что подобная ситуация возникнет и в другой реализации. По-видимому, большое различие в трудоемкости 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 завершения алгоритма.

Целью качественного анализа, представленного в табл. 4, является сравнение возможностей, которые предоставляют другие интервальные библиотеки. Для этого рассмотрим подробнее их функциональность, точнее, такие ее аспекты, как возможность работы с различными интервальными арифметиками, возможность выполнения матричных операций, наличие гото-

вых 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 4

Качественное сравнение интервальных библиотек

Table 3

Qualitative Comparison of Interval Libraries

Библиотека	Арифм.	Мат. операции	Визуал.	Оценка парам.	Решение уравнений	Оптим.
IntvalPy	+	+	+	+	+	+
PyInterval	+	–	–	–	+	–
PyIbex	–	+	+	–	+	+
Octave[2]	–	+	–	+	+	–
WM[3]	+	+	–	+	–	–
JInterval	+	+	+	+	+	+
JI[4]	–	+	+	–	+	+
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.

Заключение

В работе были изложены особенности разработанной библиотеки IntvalPy, главные идеи, положенные в ее основу, а также ее 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 определенных 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 нее добавляется новый инструментарий.

Список литературы

1. **Андросов А. С., Шарый С. П.** Библиотека IntvalPy. Программный репозиторий // URL: https://github.com/AndrosovAS/intvalpy
2. **Kearfott R. B., Nakao M., Neumaier A., Rump S. M. et al.** Standardized notation in interval analysis // Reliable Computing. 2010, vol. 15, no. 1, pp. 7–13.
3. **Rump S. M.** Verification methods: Rigorous results using floating-point arithmetic // Acta Numerica. 2010, vol. 19, pp. 287–449.

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

[3] Интервальные возможности проприетарной системы Wolfram Mathematica.

[4] JuliaIntervals.

4. **Хансен Э., Уолстер Дж. У.** Глобальная оптимизация с помощью методов интервального анализа // Институт компьютерных исследований. М., 2012, 516 с.
5. **Sharaya I. A.** Boundary intervals and visualization of AE-solution sets for interval systems of linear equations // Reliable Computing. 2012, pp. 435–467.
6. **Шарая И. А.** Метод граничных интервалов для визуализации полиэдральных множеств решений // Вычислительные технологии. 2015, Т. 20, С. 75–103.
7. **Шарый С. П.** Разрешимость интервальных линейных уравнений и анализ данных с неопределенностями // Автоматика и телемеханика. 2012, С. 111–125.
8. **Шарый С. П.** Сильная согласованность в задаче восстановления зависимостей при интервальной неопределенности данных // Вычислительные технологии. 2017, Т. 22, С. 150–172.
9. **Fiedler M., Nedoma J., Ramik J., Rohn J., Zimmermann K.** Linear optimization problems with inexact data //Springer Science+Business Media. New York, 2006, 223 p.
10. **Шарый С. П.** Конечномерный интервальный анализ // Новосибирск: XYZ, 2022, 654 с.
11. **Neumaier A.** Interval methods for systems of equations // Cambridge. New York, 1990, 270 p.
12. **Стецюк П. И.** Субградиентные методы ralgb5 и ralgb4 для минимизации овражных выпуклых функций // Вычислительные технологии. 2017, Т. 22, С. 127–149.
13. **Shary S. P.** On optimal solution of interval linear equations // SIAM Journal on Numerical Analysis. 1995, pp. 610–630.
14. **Шарая И. А.** Пакет IntLinIncR3. Руководство пользователя // 2014, 26 с, URL: http://www.nsc.ru/interval/Programing/MCodes/IntLinIncR3.pdf
15. **Ostanina T. N., Rudoi V. M., Patrushev A. V., Darintseva A. B., Farlenkov A. S.** Modelling the dynamic growth of copper and zinc dendritic deposits under the galvanostatic electrolysis conditions // Journal of Electroanalytical Chemistry. 2015, vol. 750, pp. 9–18.
16. **Ostanina T. N., Rudoy V. M., Nikitin V. S., Darintseva A. B., Zalesova O. L., Porotnikova N. M.** Determination of the surface of dendritic electrolytic zinc powders and evaluation of its fractal dimension // Russian Journal of Non-Ferrous Metals. 2017, vol. 57, pp. 47–51.
17. **Баженов А. Н., Жилин С. И., Кумков С. И., Шарый С. П.** Обработка и анализ данных с интервальной неопределенностью // 2022, 242 с, URL: http://www.nsc.ru/interval/Library/ApplBooks/InteDataProcessing.pdf
18. **Kumkov S. I., Nikitin V. S., Ostanina T. N., Rudoy V. M.** Interval processing of electrochemical data // Computational and Applied Mathematics. 2020, vol. 380, 7 p.
19. **Shary S. P.** Graph subdivision methods in interval global optimization // Studies in Computational Intelligence. 2014, vol. 539, pp. 153–170.
20. **Taschini S.** Interval Arithmetic: Python Implementation and Applications // Proceedings of the 7th Python in Science Conference (SciPy 2008). 2008, pp. 16–22.
21. **Hedar A. R., Ahmed A.** Studies on metaheuristics for continuous global optimization problems // Kyoto University, Kyoto, Japan. 2004, 148 p.
22. **Nadezhin D. Yu., Zhilin S. I.** JInterval Library: Principles, Development, and Perspectives // Reliable Computing. 2014, vol. 19, no. 3, pp. 229–247.
23. **Shary S. P.** Interval regularization for imprecise linear algebraic equations // 2018, 21 p., URL: https://www.researchgate.net/publication/328063512_Interval_regularization_for_imprecise_linear_algebraic_equations
24. **Oettli W., Prager W.** Compatibility of approximate solution of linear equations with given error bounds for coefficients and right-hand sides // Numerische Mathematik. 1964, vol. 6, pp. 405–409.
25. **Jr D. M., Tortelli L. M., Finger A. F., Loret A. B.** KaucherPy: interval package for Kaucher arithmetic // 2019, 12 p.

References

1. **Androsov A. S., Shary S. P.** The IntvalPy library [Online]. URL: https://github.com/AndrosovAS/intvalpy.
2. **Kearfott R. B., Nakao M., Neumaier A., Rump S. M. et al.** Standardized notation in interval analysis. *Reliable Computin*g, 2010, vol. 15, no. 1, pp. 70–13.
3. **Rump S. M.** Verification methods: Rigorous results using floating-point arithmetic. *Acta Numerica*, 2010, vol. 19, pp. 287–449.
4. **Hansen E., Walster G. W.** Global Optimization Using Interval Analysis: Revised And Expanded CRC Press; 2nd edition (December 19, 2003). 2003, 728 p.
5. **Sharaya I. A.** Boundary intervals and visualization of AE-solution sets for interval systems of linear equations. *Reliable Computing*, 2012, pp. 435–467.
6. **Sharaya I. A.** Boundary interval method for visualization of polyhedral sets of solutions. *Reliable Computin*g, 2015, vol. 20, pp. 75–103. (in Russ.)
7. **Shary S. P.** Solvability of interval linear equations and data analysis with uncertainties. *Automatics and Telemechanic*s, 2012, pp. 111–125. (in Russ.).
8. **Shary S. P.** Strong compatibility in the problem of dependency recovery under interval uncertainty. *Reliable Computing*, 2017, vol. 22, no. 2, pp. 150–172. (in Russ.).
9. **Fiedler M., Nedoma J., Ramik J., Rohn J., Zimmermann K.** Linear optimization problems with inexact data Springer Science+Business Media. New York, 2006. 223 p.
10. **Shary S. P.** Finite-dimensional interval analysis Novosibirsk: Publishing «XYZ», 2022. 654 p. (in Russ.).
11. **Neumaier A.** Interval methods for systems of equations Cambridge. New York, 1990. 270 p.
12. **Stetsyuk P. I.** Subgradient methods ralgb5 and ralgb4 for minimizing convex functions. *Reliable Computing*, 2017, vol. 22, no. 2, pp. 127--149. (in Russ.).
13. **Shary S. P.** On optimal solution of interval linear equations. *SIAM Journal on Numerical Analysi*s, 1995, pp. 610–630.
14. **Sharaya I. A.** The IntLinIncR3 package. User's Guide [Online]. 2014. 26 p. URL: http://www.nsc.ru/interval/Programing/MCodes/IntLinIncR3.pdf. (in Russ.)
15. **Ostanina T. N., Rudoi V. M., Patrushev A. V., Darintseva A. B., Farlenkov A. S.** Modelling the dynamic growth of copper and zinc dendritic deposits under the galvanostatic electrolysis conditions. *Journal of Electroanalytical Chemistry*, 2015, vol. 750, pp. 9–18.
16. **Ostanina T. N., Rudoy V. M., Nikitin V. S., Darintseva A. B., Zalesova O. L., Porotnikova N. M.** Determination of the surface of dendritic electrolytic zinc powders and evaluation of its fractal dimension Russian. *Journal of Non-Ferrous Metals*, 2017, vol. 57, pp. 47–51.
17. **Bazenov A. N., Zhilin S. I., Kumkov S. I., Shary S. P.** Processing and analysis of data with interval uncertainty [Online]. 2022, 242 p. URL: http://www.nsc.ru/interval/Library/ApplBooks/InteDataProcessing.pdf. (in Russ.)
18. **Kumkov S. I., Nikitin V. S., Ostanina T. N., Rudoy V. M.** Interval processing of electrochemical data. *Computational and Applied Mathematics*, 2020, vol. 380, 7 p.
19. **Shary S. P.** Graph subdivision methods in interval global optimization. *Studies in Computational Intelligence*, 2014, vol. 539, pp. 153–170.
20. **Taschini S.** Interval Arithmetic: Python Implementation and Applications. Proceedings of the 7th Python in Science Conference (SciPy 2008), 2008, pp. 16–22.
21. **Hedar A. R., Ahmed A.** Studies on metaheuristics for continuous global optimization problems. Kyoto: Kyoto University, 2004. 148 p.
22. **Nadezhin D. Yu., Zhilin S. I.** JInterval Library: Principles, Development, and Perspectives. *Reliable Computing*, 2014, vol. 19, no. 3, pp. 229–247.

23. **Shary S. P.** Interval regularization for imprecise linear algebraic equations [Online]. 2018. 21 p. URL: https://www.researchgate.net/publication/328063512_Interval_regularization_for_imprecise_linear_algebraic_equations.
24. **Oettli W., Prager W.** Compatibility of approximate solution of linear equations with given error bounds for coefficients and right-hand sides. *Numerische Mathematik*, 1964, vol. 6, pp. 405–409.
25. **Jr D. M., Tortelli L. M., Finger A. F., Loret A. B.** KaucherPy: interval package for Kaucher arithmetic. 2019. 12 p.

Информация об авторах

Артём Станиславович Андросов, аспирант

Сергей Петрович Шарый, доктор физико-математических наук

Information about the Authors

Androsov Artem Stanislavovich, postgraduate student

Shary Sergey Petrovich, Doctor of Sciences (Physics and Mathematics)

Статья поступила в редакцию 03.12.2022;
одобрена после рецензирования 16.01.2023; принята к публикации 16.01.2023

The article was submitted 03.12.2022 ;
approved after reviewing 16.01.2023; accepted for publication 16.01.2023