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 Прикладная математика и информатика

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Атаманова Александра Сергеевна

Тема работы: Создание руководства пользователя интервальным расширением системы компьютерной математики Octave

«К защите допущен»

Заведующий кафедрой,

д. ф.-м. н., проф.

Шарый С.П. / _____

(Фамилия И.О.) (Подпись)

« . . » . . . . . 20 . . г.

Научный руководитель

д. ф.-м. н., проф.

в. н. с., ИВТ СО РАН

Шарый С.П. / _____

(Фамилия И.О.) (Подпись)

« . . » . . . . . 20 . . г.

Дата защиты: « . . » . . . . . 20 . . г.

Новосибирск, 2019

Содержание

1 Введение

В реальной жизни, в отличии от теоретических выкладок, мы не всегда можем работать с точными значениями необходимых нам величин. Порой известны лишь границы их изменения. Это подводит человека к созданию математического аппарата для работы с интервалами, и им стал интервальный анализ, возникший в середине прошлого столетия.

Для начала мы разберём, что такое интервал и интервальный анализ, а также как работать с интервальными величинами в системе компьютерной математики Octave.

Интервал — это замкнутый отрезок вещественной оси (в комплексном случае конструкция сложнее), и, как уже говорилось, в концепции интервального анализа мы работаем не с вещественными (комплексными) числами, а с вещественными (комплексными) интервалами.

Стоит отметить, что в процессе вычислений на ЭВМ мы сталкиваемся с погрешностью. При работе с интервальными переменными мы знаем область изменения рассматриваемой величины. При выполнении некоторых необходимых операций над интервалом мы получаем, в каких пределах находится интересующая нас переменная. Это актуально во многих сферах деятельности, например, в задачах, связанных с допустимыми нагрузками, где важно знать приемлемые отклонения определённых величин.

Большим плюсом работы с интервалами является то, что ошибка вычисления накапливается в самом ответе. Эта осо-

бенность может быть использована в пользу исследователя.

GNU Octave — свободно распространяемая система для математических вычислений. Она предназначена, в первую очередь, для численных расчётов, но также характеризуется мощным математически ориентированным синтаксисом со встроенными инструментами визуализации и математического вычисления. Она является популярной средой программирования со своим высокоуровневым языком, близким к языку MATLAB. Интервальный пакет для Octave обеспечивает интервальное расширение системы. Он создан Оливером Хаймлихом (Германия) и помимо собственно интервальной арифметики и элементарных интервальных операций включает некоторые процедуры для решения ряда стандартных задач интервального анализа. Распространяется свободно и является одним из программных средств, удовлетворяющих стандарту IEEE-1788 на интервальные вычисления на ЭВМ. А таких существует не так много на данный момент.

Разработка интервального пакета Octave началась в сентябре 2014 года, и уже в январе 2015 года был готов первый выпуск пакета, который содержал полный набор функций, требуемых стандартом. Сам стандарт IEEE Std 1788-2015 был разработан в июле 2014 года и был внесён на рассмотрение. И уже 11 июня 2014 года стандарт был утвержден. Сейчас же интервальный пакет Octave является законченным продуктом, который может быть использован для практических нужд интервального анализа.

Существует достаточно литературы по работе в Octave, но

на русском языке нет руководства по использованию её интервального расширения. Это и вызвало необходимость моей работы.

Целью моей работы является написание руководства пользователя по использованию интервального расширения Octave. Оно будет включать описание базовой теории по интервальному анализу, основных принципов работы с интервальным пакетом, а также реализация некоторых интервальных методов.

2 Основные понятия интервального анализа

Интервал вещественной оси $[x, y]$ — это множество чисел, расположенных между заданными числами x и y, включая их самих, где x — левая граница интервала, а y — правая.

$$[x, y] = \{\xi \in \mathbb{R} \mid x \leq \xi \leq y\}.$$

В соответствии со стандартом IEEE-1788 интервалы (интервальные величины) выделяются математическим жирным шрифтом $\boldsymbol{x} = [\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}}]$, где нижняя черта над интервалом будет обозначать его нижнюю границу, а верхняя, соответственно, верхнюю. В дальнейшем будем придерживаться этих обозначений. Стоит отметить, что если левая и правая границы интервала совпадают, то его называют точечным (вырожденным) интервалом: $\boldsymbol{x} = [x, x]$.

Интервал $\boldsymbol{x}$ называется правильным, если $\overline{\boldsymbol{x}} \geq \underline{\boldsymbol{x}}$. Если же $\overline{\boldsymbol{x}} < \underline{\boldsymbol{x}}$, тогда интервал называется неправильным. В данной работе мы рассматриваем только правильные, поскольку только они реализованы в Octave. Есть теории и арифметики, которые работают также и с неправильными интервалами, но их описание мы опустим.

Введём операции над интервалами, чтобы определить интервальную арифметику для дальнейшей формализации работы с интервалами. Существует много способов введения операций над интервалами. В зависимости от конкретной задачи стоит выбирать тот, который больше соответствует её нуждам. Мы же рассмотрим простой, естественный принцип

введения операций над интервалами

$$\boldsymbol{x} * \boldsymbol{y} = \{x * y \mid x \in \boldsymbol{x}, y \in \boldsymbol{y}\},$$

где $* \in \{+, -, \cdot, /\}$.

Приведём развёрнутые формулы операций над интервалами:

$$\begin{aligned}
\boldsymbol{x} + \boldsymbol{y} &= \left[\underline{\boldsymbol{x}} + \underline{\boldsymbol{y}}, \overline{\boldsymbol{x}} + \overline{\boldsymbol{y}}\right], \\
\boldsymbol{x} - \boldsymbol{y} &= \left[\underline{\boldsymbol{x}} - \underline{\boldsymbol{y}}, \overline{\boldsymbol{x}} - \overline{\boldsymbol{y}}\right], \\
\boldsymbol{x} \cdot \boldsymbol{y} &= \left[\min\left\{\underline{\boldsymbol{x}}\underline{\boldsymbol{y}}, \underline{\boldsymbol{x}}\overline{\boldsymbol{y}}, \underline{\boldsymbol{y}}\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}\overline{\boldsymbol{y}}\right\}, \max\left\{\underline{\boldsymbol{x}}\underline{\boldsymbol{y}}, \underline{\boldsymbol{x}}\overline{\boldsymbol{y}}, \underline{\boldsymbol{y}}\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}\overline{\boldsymbol{y}}\right\}\right], \\
1/\boldsymbol{x} &= \left[1/\overline{\boldsymbol{x}}, 1/\underline{\boldsymbol{x}}\right], \text{ если } \ 0 \notin \boldsymbol{x}, \\
\boldsymbol{x}/\boldsymbol{y} &= \boldsymbol{x} \cdot 1/\boldsymbol{y}.
\end{aligned}$$

Деление одного интервала на другой нульсодержащий интервал не определено. В дальнейшем мы будем сталкиваться с необходимостью выполнения этого действия. По большому счёту из-за одной точки нуль в интервале мы не можем выполнить операцию. Снимем это ограничение, чтобы определить так называемую расширенную интервальную арифметику Кахана $< \mathbb{K}\mathrm{h}\mathbb{R}, +, -, \cdot, / >$, деление в которой определяется следующим образом:

$$\boldsymbol{x}/\boldsymbol{y} = \begin{cases} \left[\overline{\boldsymbol{x}}/\underline{\boldsymbol{y}}, \infty\right], & \text{если } \overline{\boldsymbol{x}} \leq 0 \text{ и } \underline{\boldsymbol{y}} = 0, \\ [-\infty, \overline{\boldsymbol{x}}/\overline{\boldsymbol{y}}] \cup \left[\overline{\boldsymbol{x}}/\underline{\boldsymbol{y}}, \infty\right], & \text{если } \overline{\boldsymbol{x}} \leq 0 \text{ и } \underline{\boldsymbol{y}} < 0 < \overline{\boldsymbol{y}}, \\ [-\infty, \overline{\boldsymbol{x}}/\overline{\boldsymbol{y}}], & \text{если } \overline{\boldsymbol{x}} \leq 0 \text{ и } \underline{\boldsymbol{y}} = 0, \\ [-\infty, +\infty], & \text{если } \underline{\boldsymbol{x}} < 0 < \overline{\boldsymbol{x}}, \\ \left[-\infty, \underline{\boldsymbol{x}}/\underline{\boldsymbol{y}}\right], & \text{если } \underline{\boldsymbol{x}} \geq 0 \text{ и } \overline{\boldsymbol{y}} = 0, \\ \left[-\infty, \underline{\boldsymbol{x}}/\underline{\boldsymbol{y}}\right] \cup [\underline{\boldsymbol{x}}/\overline{\boldsymbol{y}}, \infty], & \text{если } \underline{\boldsymbol{x}} \geq 0 \text{ и } \underline{\boldsymbol{y}} < 0 < \overline{\boldsymbol{y}}, \\ [\underline{\boldsymbol{x}}/\overline{\boldsymbol{y}}, \infty], & \text{если } \underline{\boldsymbol{x}} \geq 0 \text{ и } \underline{\boldsymbol{y}} = 0. \end{cases}$$

Стоит отметить, что есть разные арифметики для работы с интервальными величинами. Поскольку в Octave реализована арифметика Кахана (хоть и с некоторыми поправками, о которых в разделе «Соответствие стандарту»), далее будет говориться про неё.

Для определённости имеет смысл выписать, как правильно работать с бесконечными и полубесконечными интервалами:

$$[\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}}] + [-\infty, \overline{\boldsymbol{y}}] = [-\infty, \overline{\boldsymbol{x}} + \overline{\boldsymbol{y}}],$$
$$[\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}}] + \left[\underline{\boldsymbol{y}}, \infty\right] = \left[\underline{\boldsymbol{x}} + \underline{\boldsymbol{y}}, \infty\right],$$
$$[\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}}] + [-\infty, \infty] = [-\infty, \infty],$$
$$[\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}}] - [-\infty, \infty] = [-\infty, \infty],$$
$$[\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}}] - [-\infty, \overline{\boldsymbol{y}}] = [\underline{\boldsymbol{x}} - \overline{\boldsymbol{y}}, \infty],$$
$$[\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}}] - \left[\underline{\boldsymbol{y}}, \infty\right] = \left[-\infty, \overline{\boldsymbol{x}} - \underline{\boldsymbol{y}}\right].$$

Операции умножения и деления с ними осуществляются нечасто, поэтому результаты работы с ними не были описаны.

Рассмотрим, какими же характеристиками можно описы-

вать интервальные величины. Интервал может быть задан полностью его концами. Также для его описания можно ввести следующие характеристики:

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

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

$$\operatorname{wid} \boldsymbol{x} = \overline{\boldsymbol{x}} - \underline{\boldsymbol{x}} \text{ — ширина интервала.}$$

Также введем понятия магнитуды (модуля) и мигнитуды интервала. Магнитуда — это наибольшее из значений интервала:

$$|\boldsymbol{a}| := \max\{|a| \mid a \in \boldsymbol{a}\} = \max\left\{|\overline{\boldsymbol{a}}|, |\underline{\boldsymbol{a}}|\right\}.$$

А мигнитуда — это наименьшее из абсолютных значений точек интервала:

$$\langle \boldsymbol{x} \rangle := \min\{|x| \mid x \in \boldsymbol{x}\} = \begin{cases} \max\left\{|\overline{\boldsymbol{x}}|, |\underline{\boldsymbol{x}}|\right\}, & \text{если } 0 \notin \boldsymbol{x}, \\ 0, & \text{если } 0 \in \boldsymbol{x}. \end{cases}$$

Рассмотрим всем известный дистрибутивный закон для сложения и умножения $x(y+z) = xy+xz$ для любых $x, y, z \in \mathbb{R}$. Однако в интервальном анализе с введенными нами операциями в общем случае он не выполняется. Но справедливо включение, которое называется законом субдистрибутивности

$$\boldsymbol{x}(\boldsymbol{y} + \boldsymbol{z}) \subseteq \boldsymbol{x}\boldsymbol{y} + \boldsymbol{x}\boldsymbol{z}.$$

Отметим, что интервалы — это множества, поэтому операция включения понимается в теоретико-множественном смыс-

ле.

Еще одно правило не распространяется на интервальную арифметику. В данной теории

$$\boldsymbol{x} - \boldsymbol{x} \neq 0.$$

Это происходит из-за того, что при вычитании рассчитывается интервал, содержащий разность между всеми возможными результатами двух независимых чисел, лежащих внутри $\boldsymbol{x}$, а не разность между двумя одинаковыми числами.

Важным результатом в интервальном анализе является основная теорема интервальной арифметики.

Теорема 1 . *Пусть $f(x_1, \ldots, x_n)$ — рациональная вещественная функция аргументов $x_1, \ldots, x_n$ и для неё определен результат $\boldsymbol{f}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ подстановки вместо аргументов интервалов их изменения $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{IR}$ и выполнений всех действий над ними по правилам интервальной арифметики. Тогда*

$$\{\, f(x_1, \ldots, x_n) \mid x_1 \in \boldsymbol{x}_1, \ldots, x_n \in \boldsymbol{x}_n \} \subseteq \boldsymbol{f}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n),$$

то есть $\boldsymbol{f}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ содержит множество значений функции $f(x_1, \ldots, x_n)$ на $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$. Если выражение для $f(x_1, \ldots, x_n)$ содержит не более чем по одному вхождению каждой переменной в первой степени, то вместо включения выполняется точное равенство.

Теперь перейдём к рассмотрению понятий интервальных векторов и матриц.

Если $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ — это интервалы, то вектор-столбцом будем называть

$$\boldsymbol{a} = \begin{pmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \vdots \\ \boldsymbol{a}_n \end{pmatrix}$$

и соответственно вектор-строкой

$$\boldsymbol{a} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n),$$

где $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ являются компонентами вектора.

Интервальные вектора геометрически можно интерпретировать как прямоугольные параллепипеды со сторонами, параллельными осям координат. В дальнейшем мы их будем называть брусами.

Если S — непустое ограниченное множество в $\mathbb{R}^n$, то его интервальной оболочкой $\square S$ называется наименьший по включению интервальный вектор, содержащий S, то есть $\square S$ — это пересечение всех интервальных векторов, содержащих S. Можно сказать, что она является интервальным объектом, наилучшим образом внешне приближающий заданное множество.

Перейдём к рассмотрению интервальных матриц. Интервальной матрицей называется прямоугольная матрица, компонентами которой являются интервалы:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{a}_{11} & \boldsymbol{a}_{12} & \dots & \boldsymbol{a}_{1n} \\ \boldsymbol{a}_{21} & \boldsymbol{a}_{22} & \dots & \boldsymbol{a}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{a}_{m1} & \boldsymbol{a}_{m2} & \dots & \boldsymbol{a}_{mn} \end{pmatrix}$$

Обозначаться они будут следующим образом $\boldsymbol{A} = (\boldsymbol{a}_{ij})$.

Можно сказать, что интервальные векторы являются интервальными матрицами размера $n \times 1$ или $1 \times n$.

Неособенной интервальной матрицей $\boldsymbol{A}$ называется квадратная интервальная матрица, такая что $\forall A \in \boldsymbol{A}\ \det A \neq 0$.

Такие операции как wid ($\boldsymbol{A}$), rad ($\boldsymbol{A}$) и mid ($\boldsymbol{A}$) применяются к матрице покомпонентно. Теперь определим сложение, вычитание и умножения.

В одномерном случае интервалов произведение определялось как множество всевозможных произведение между точечными представителями (точками вещественной оси). В силу вычислительных трудностей умножение в многомерном случае выполнять операцию между каждым точечным представителям является весьма трудоёмкой задачей.
Поэтому введём операции между матрицами другим образом:

$$\boldsymbol{A} * \boldsymbol{B} = \square\{A * B \mid A \in \boldsymbol{A}, B \in \boldsymbol{B}\},$$

где $* \in \{+, -, \cdot\}$. Исходя из этого, выпишем операции над векторами и матрицами.

Сумма (разность) двух интервальных матриц одного размера есть матрица того же размера, полученная поэлемент-

ным сложением (вычитанием) операндов:

$$\boldsymbol{A} \pm \boldsymbol{B} = \begin{pmatrix} \boldsymbol{a}_{11} \pm \boldsymbol{b}_{11} & \boldsymbol{a}_{12} + \boldsymbol{b}_{12} & \dots & \boldsymbol{a}_{1n} \pm \boldsymbol{b}_{1n} \\ \boldsymbol{a}_{21} \pm \boldsymbol{b}_{21} & \boldsymbol{a}_{22} \pm \boldsymbol{b}_{22} & \dots & \boldsymbol{a}_{2n} \pm \boldsymbol{b}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{a}_{m1} \pm \boldsymbol{b}_{m1} & \boldsymbol{a}_{m2} \pm \boldsymbol{b}_{m2} & \dots & \boldsymbol{a}_{mn} \pm \boldsymbol{b}_{mn} \end{pmatrix}$$

Если $\boldsymbol{A} = (\boldsymbol{a}_{ij}) \in \mathbb{K}\mathrm{h}\mathbb{R}^{m \times l}$ и $B = (\boldsymbol{b}_{ij}) \in \mathbb{K}\mathrm{h}\mathbb{R}^{l \times n}$, то произведение матриц A и B есть матрица $C = (\boldsymbol{c}_{ij}) \in \mathbb{K}\mathrm{h}\mathbb{R}^{m \times n}$, такая что

$$\boldsymbol{c}_{ij} = \sum_{k=1}^{l} \boldsymbol{a}_{ik} \boldsymbol{b}_{kj}.$$

Только для одного частного случая произведение матриц является множеством всевозможных точечных произведений представителей, когда интервальная матрица $\boldsymbol{A} \in \mathbb{K}\mathrm{h}\mathbb{R}^{m \times l}$ умножается на вектор $b \in \mathbb{R}^l$:

$$\boldsymbol{A}b = \{Ab \mid A \in \boldsymbol{A}\}.$$

Сложение — это ассоциативная и коммутативная операция, но умножение таковой не является. Поэтому при умножении матриц нужно учитывать порядок расстановки скобок сомножителей. Но некоторые полезные факты про частные случаи умножения мы можем сказать.

1. Если $\boldsymbol{A} = (\boldsymbol{a}_{ij}) \in \mathbb{K}\mathrm{h}\mathbb{R}^{m \times l}$, $B = (\boldsymbol{b}_{ij}) \in \mathbb{R}^{l \times k}$, $\boldsymbol{C} = (\boldsymbol{c}_{ij}) \in \mathbb{R}^{l \times n}$, то
$$\boldsymbol{A}(BC) \subseteq (\boldsymbol{A}B)C;$$

2. Если $\boldsymbol{A} = (\boldsymbol{a}_{ij}) \in \mathbb{R}^{m \times l}$, $B = (\boldsymbol{b}_{ij}) \in \mathbb{K}\mathrm{h}\mathbb{R}^{l \times k}$, $\boldsymbol{C} = (\boldsymbol{c}_{ij}) \in$

$\mathbb{R}^{l\times n}$, то

$$(A\boldsymbol{B})C = A(\boldsymbol{B}C).$$

Перейдём к рассмотрению интервальной обратной матрицы. Существует несколько способов её введения. В неинтервальном случае под обратной матрицей часто понимают матрицу A^{-1} такую, что $AA^{-1} = I$, где I — единичная матрица. Но переложить это определение на интервальный случай будет не совсем разумной идеей, поскольку во введённой нами арифметике не действует правило ассоциативности. Поэтому, например, при решении линейной системы интервальных уравнений следующие выкладки неверны ($\boldsymbol{I}$ — интервальная единичная матрица):

$$\boldsymbol{A}x = \boldsymbol{b}$$
$$\boldsymbol{A}^{-1}(\boldsymbol{A}x) = \boldsymbol{A}^{-1}\boldsymbol{b}$$
$$(\boldsymbol{A}^{-1}\boldsymbol{A})x = \boldsymbol{A}^{-1}\boldsymbol{b}$$
$$\boldsymbol{I}x = \boldsymbol{A}^{-1}\boldsymbol{b}$$
$$x = \boldsymbol{A}^{-1}\boldsymbol{b}$$

Предложим следующее определение обратной интервальной матрицы, и, как в точечном случае, для неособенных матриц. Для матрицы $\boldsymbol{A} \in \mathbb{K}\mathrm{h}\mathbb{R}^n$ обратной интервальной матрицей будем называть

$$\boldsymbol{A}^{-1} = \square\{A^{-1} \mid A \in \boldsymbol{A}\}.$$

Получается, что обратная матрица является интервальной оболочкой множество обратных для точечных матриц, содер-

жащихся в $\boldsymbol{A}$.

Мы рассматриваем только вещественный случай интервалов, но существует также и комплексный. Здесь уже интервал является двумерным, представленный своей действительной и мнимой частью.

Существуют несколько способов для представления комплексных интервалов. Часто рассматриваются два: в виде прямоугольников и в виде кругов. Прямоугольным комплексным интервалом $\boldsymbol{a}+i\boldsymbol{b}$, где $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{KR}$, называется множество комплексной плоскости

$$\{z = a + ib \in \mathbb{C} \mid a \in \boldsymbol{a}, b \in \boldsymbol{b}\}.$$

Круговым комплексным интервалом $< c, r >$, $c \in \mathbb{C}$, $r \in \mathbb{R}$ и $r > 0$, называется множество комплексной плоскости

$$\{z \in \mathbb{C} \mid |z - c| \leq r\}.$$

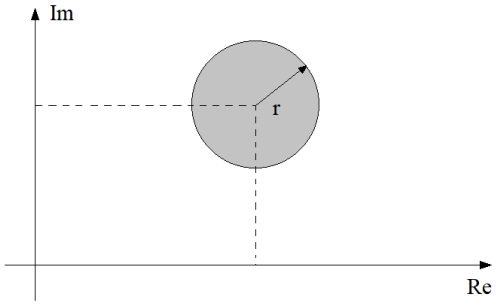

Рис. 1: Круговой комплексный интервал

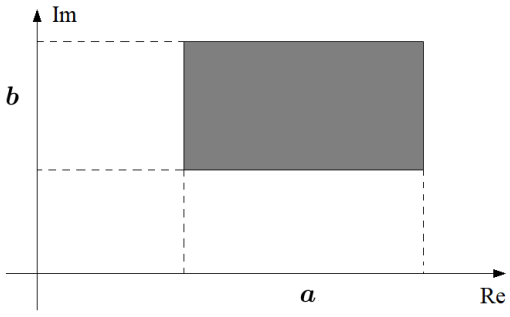

Рис. 2: Прямоугольный комплексный интервал

В Octave комплексные интервалы не реализованы, поэтому ограничимся лишь их упоминанием.

3 Octave

3.1 Начало работы

Octave является свободно распространяемым программным обеспечением. На официальном сайте

`https://www.gnu.org/software/octave/download.html`,

его можно скачать в зависимости от ОС, которой Вы пользуетесь (OS Linux, OS Windows), а также выбрать все нужные для пользования пакеты. В данной работе нас интересует пакет интервальной арифметики. При каждом запуске Octave необходимо подгрузить интервальный пакет следующим образом

```
>> pkg load interval
```

Или же можно прописать загрузку пакета в сценарии начальной конфигурации, чтобы каждый раз не писать эту команду. При необходимости можно вызывать команду `help`, чтобы подробнее узнать о возможностях, аргументах и других особенностях работы с интересующей нас функцией:

```
>> help infsupdec
>> help wid
```

Вам предоставляется вся встроенная информация о введённой вами функции, а также ссылки, где находится подробности работы с ней.

Опишем фундаментальную концепцию интервального пакета.

1. Интервалы — замкнутые связные подмножества $\mathbb{R}$. Они могут быть ограничены или неограничены, а также пустыми. $-\inf$ и $+\inf$ являются соответственно левой или правой границей неограниченного интервала.

2. Определены функции от интервала следующем способом: результат функции в общем случае является интервалом, содержащем все значения соответствующей ей точечной функции, где она определена.

3. Не реализована работа с мультиинтервалами. Про это подробнее в следующем разделе.

4. Реализованы высокоуровневые функции для работы с интервальными величинами.

3.2 Особенности арифметики

Скажем про существование интервальной арифметики Каухера, которая была получена путем введения обратных по сложению и умножению. Она является алгебраически полной. Но не каждая арифметика обладает «хорошим» свойством алгебраической полноты, которое позволяет нам свободно работать в ней. Вообще говоря, для большинства практических задач полнотой арифметики можно пренебречь.

В одном из предыдущих разделов я описала так называемую расширенную интервальную арифметику, которая носит имя Кахана. Она реализована на Octave в усечённом виде. При делении на интервал, содержащий нуль, по правилам арифметики Кахана мы должны получить так называемый

мультиинтервал (объединение нескольких непересекающихся интервалов). Но при рассмотрении последующего примера мы видим, что это не так:

```
>> pkg load interval
>> a=infsup(1);
>> b=infsup(-1,1);
>> a/b
ans=[Entire]
```

Под `Entire` здесь понимается вся вещественная ось $\mathbb{R}$.

Как вы могли уже заметить, при выполнении операций, система выдает нам её результат. Если же мы хотим его скрыть, то необходимо ставить «;» в конце команды.

Для простоты в интервальном пакете Octave не была реализована работа с мультиинтервалами. Мультиинтервалом называется конечное объединение непересекающихся интервалов. В рассматриваемом примере мы убедились, что интервал $]-1,+1[$ (где «обратные» скобки означают, что граница интервала не принадлежит ему), который должен быть исключён из рассмотрения, мы включаем в ответ.

Таким образом в интервальном пакете Octave реализованы не в точности расширенная арифметика Кахана, а её урезанная версия. Несмотря на все эти допущения, касаемые арифметики, мы имеем технический аппарат для практических нужд интервального анализа.

3.3 Округление

Машинно представимые числа на ЭВМ (числа с плавающей запятой) — это вещественные числа, которые в памяти компьютера могут быть представленны с помощью мантиссы и показателя степени с определённой точностью (float или double). Если рассматривать вещественную прямую, то на ней их можно отметить. Также есть и машинно непредставимые числа. Округление происходит после каждой операции, тем самым накапливая некоторую погрешность вычисления. При выполнения действий над интервалами мы получаем новые интервалы.

Интервал можно задать его концами, и может оказаться, что они не являются машинно представимыми числами. Тогда стоит решить, в какую «сторону» производить округление.

В Octave реализована технология внешнего направленного округления, то есть левый конец результирующего интервала округляется до машинно представимого числа в направлении $-\infty$, а случае правого конца в направлении $+\infty$. Таким образом у нас происходит расширение интервала, что негативно сказывается на точности полученных результатов. Но с другой стороны мы избегаем ситуации, когда при сужении интервала теряется какие-то значения, необходимые нам. Этот «плохой» случай может произойти при округлении к ближайшему машинно представимому числу. Все рассуждения могут быть перенесены на случай многомерных интервалов, а также на комплексный случай.

Интервальный пакет реализован таким образом, что точность вычислений выше, чем в ядре самой Octave. В силу этого, можно расчитывать на получение точных адекватных результатов.

3.4 Задание интервалов

Перед тем как приступать к работе, мы определим, как можно задавать интервалы в Octave. Существуют несколько способов записи интервальных величин:

1. Задание левой и правой границы с помощью конструктора `infsup` и `infsupdec`. В качестве первого аргумента задаётся левая граница интервала, а в качестве второго — правая. Если это вырожденный интервал, совпадающий с вещественным числом, прописывается только 1 аргумент.

```
>> infsupdec(0,1)
ans=[0,1]_com

>> infsup(1)
ans=[1,1]
```

2. Задание радиуса (первый аргумент) и середины (второй аргумент) интервала.

```
>>midrad (12, 3)
ans=[9, 15]_com
```

3. Наименьший по включению интервал, содержащий данный аргумент.

```
>>infsupdec ("1.1")
ans=[1.0999, 1.1001]_com
```

4. Задаётся такой наименьший по включению интервал, в который входят все числа, записанные в аргументе.

```
>>hull (3, 42, "19.3", "-2.3")
ans=[-2.3001, +42]_com
```

При попытки ввести неправильный интервал, система будет выдавать ошибку.

Стоит отметить, что символьные константы и числа с плавающей запятой в вышеперечисленных примерах передаются в конструктор в виде строки (заключенные в двойные кавычки). При таком задании мы не получаем ошибки преобразования, которые могут произойти, когда числовой интервал преобразуется в число с плавающей точкой до того, как он подаётся в конструктор infsupdec.

Первый вариант задания:

```
infsupdec(0.3)
>>ans=[0.29999, 0.3]_com
```

Второй:

```
infsupdec("0.3")
>>ans=[0.29999, 0.30001]
```

Границы интервала хранятся в двоичном формате и преобразуются в десятеричный формат с возможной потерей точности. Как уже было сказано, при неточных входных данных

границы интервала округляются согласно оговоренным правилам округления. При конструировании интервала можно контролировать точность задания интервала, когда его границы являются числами с плавающей запятой, с помощью функции `output_precision(n)`, где `n` — это количество отображаемых знаков после запятой.

```
>> output_precision(2)
>> infsupdec(1/7,1/3)
ans = [0.14, 0.34]_com
```

С помощью функции `nextout(X)`, где `X` — это интервал, можно «расширить» его границы в каждом направлении до следующего машинно представимого числа.

```
>> x = infsup (1);
>> nextout (x) == infsup (1 - eps / 2, 1 + eps)
ans = 1
```

3.5 Операции и функции над интервалами

Над интервалами можно выполнять арифметические операции по определённым правилам, описанным в главе про интервальную арифметику.

```
>>a=infsupdec(0,1);
>>b=infsupdec(10,11);
>>a+b
ans = [10, 12]_com
>> a-b
```

```
ans = [-11, -9]_com
>> a*b
ans = [0, 11]_com
>> a/b
ans = [0, 0.10001]_com
```

Интервальный пакет даёт возможность вычислять функции от интервальных величин. В большинстве случаев их названия совпадают с названиями стандартными функциями GNU Octave. С точки системы интервалы являются множествами (подмножествами вещественной прямой).

Стоит сказать, что из-за низкоуровневой поддержки библиотек и отсутствии оптимизации операции над интервальными функциями они выполняются медленнее, чем при работе с числами с плавающей точкой.

Теперь обсудим, как понимается функция в интервальном пакете. Каждой точке интервала ставится в соответствие значение точечной функции, потом находится интервальная оболочка получившегося множества, которая и выводится нам в качестве ответа.

```
>>sin(infsupdec (0.1,0.5))
ans=[0.099833, 0.47943]_com
>>pow(infsupdec(2,3), infsupdec(8,9))
ans = [256, 19683]_com
```

Существуют также функции от интервалов, результатом которых является не интервал, а число. К таким относятся такие, как mid $\boldsymbol{a}$, wid $\boldsymbol{a}$, rad $\boldsymbol{a}$.

К интервалам можно применять булевские операции $<$, $>$, $\leq$, $\geq$, то есть производить их сравнение. Поскольку интервалы являются подмножествами вещественной оси, поэтому каждую точку одного интервала можно сравнивать с каждой точкой другого интервала. Операции сравнения позволяют получить ответ на следующий вопрос: для каждой ли точки одного интервала существует точка из другого интервала, которая лежит правее ($<$), левее ($>$), правее или сопадает ($\leq$), левее или совпадает ($\geq$) с ней. Они являются булевскими, потому выдаёт 1 (истина) или 0 (ложь).

```
>>infsupdec(0,2)<=infsupdec(1,5)
ans=1
```

К булевским функциям над интервалами можно также отнести `subset`, `isempty`, `ismember` и другие, которые будут подробнее описаны в приложении.

3.6 Соответствие стандарту

При работе с Octave можно заметить следующее. Посмотрев на результаты выполнения команд, мы можем заметить так называемые декорации, приписанные к интервалам (com, def и другие).

```
>> pkg load interval
>> infsupdec(0,1)
ans=[0,1]_com

>> infsupdec()
ans=[Empty]_trv
```

Они несут определённый смысл для пользователя. Это то, что связывает Octave со стандартом на интервальные вычисления IEEE-1788. В ходе преобразований декорации собирают дополнительную информацию об интервалах, являющихся внешним оцениванием рассматриваемой функции. Полезны они при долгой и сложной оценке функций. Например, если декорация `def` сохраняется при применении действий над интервалом и переходит в конечный результат, тогда доказано, что функция определена фактически для всех значений входных данных.

Необходимость декораций в полной мере может ощущаться при внушительном количестве операций, переменных, когда пользователю сложно следить за большим потоком данных. Более подробно с ними можно ознакомиться, собственно прочитав стандарт.

Если пользователю не нужны данные теги, то можно работать с функцией `infsup`, а не с `infsupdec`. Также можно использовать функцию `intervalpart`, чтобы снять с интервала декорацию.

```
>> infsup(-1,1)
ans=[-1,1]
```

При работе с интервальными величинами нужно следить, чтобы они были все либо декорированными, либо недекорированными. Если это не делать, то система будет выдавать ошибки.

Вообще говоря, в документации рекомендуется работать всегда с декорациями.

Ниже приведена таблица с описание декораций.

Декорация	Короткое описание	Определение
com	common	$\boldsymbol{a}$ — ограниченное непустое подмножество Dom(f), где f непрерывна в каждой точке $\boldsymbol{a}$, и вычисленный интервал f(x) ограничен
dac	defined and continuous	$\boldsymbol{a}$ — непустое подмножество Dom(f), и ограничение f на x является непрерывным
def	defined	$\boldsymbol{a}$ — непустое подмножество Dom(f)
trv	trivial	всегда верно (поэтому не дает никакой информации)
ill	ill-formed	Не интервал, хотя бы один конструктор интервала не удался в ходе вычисления

3.7 Интервальные векторы, матрицы и массивы

Octave предусматривает работу с массивами и матрицами. При подключении интервального пакета нам предоставляется возможность работы с интервальными матрицами.

Как и в случае с интервалами, интервальные матрицы задаются с помощью функции `infsupdec` (`infsup`) или `midrad`. Рассмотрим подробнее возможности задания.

1. С помощью конструктора `infsupdec`, у которого первый аргумент — матрица левых границ интервалов, которые являются её элементами, а второй аргумент — матрица правых границ интервалов.

```
>> a=[1 2;3 4];
>> b=[2 3;4 5];
>> A=infsupdec(a,b)
A = 2x2 interval matrix
[1, 2]_com   [2, 3]_com
[3, 4]_com   [4, 5]_com
```

2. Осуществляется покомпонентная запись матрицы. Элементы матрицы разделяются запятой или пробелом, а переход на следующую строку производится с помощью точки с запятой.

```
>>infsup("[1,2], 2, [3,4]; pi, e, 3")
ans = 2x3 interval matrix
           [1, 2]                  [2]   [3, 4]
 [3.1415, 3.1416]   [2.7182, 2.7183]       [3]
```

3. С помощью конструктора `midrad`, в качестве аргументов которому передаётся матрица середин интервалов и матрица с соответствующими радиусами.

```
>> m=[4 7;6 5];
>> r=[2 0;7 3];
>> midrad(m,r)
```

```
ans = 2x2 interval matrix

      [2, 6]_com      [7]_com
   [-1, +13]_com   [2, 8]_com
```

4. Для создания вектор столбца сначала задаются все левые границы интервалов, а потом через точку с запятой правые. Если же нужно задать вектор строку, то всё тоже самой, только разделяем запятой.

```
>>infsupdec ({"1.1",1,"e"};{"1.8",5.5,"e"})
ans = 1x3 interval vector
  [1.0999, 1.8001]_com
  [1, 5.5]_com
  [2.7182, 2.7183]_com
```

5. Легко можно создать единичную матрицу нужной размерности.

```
>>infsupdec(ones(3))
ans = 3x3 interval matrix
  [1]_com   [1]_com   [1]_com
  [1]_com   [1]_com   [1]_com
  [1]_com   [1]_com   [1]_com
```

6. Диагональная единичная матрица необходимой размерности задаётся следующим образом.

```
a=infsupdec(eye(3))
a = 3x3 interval matrix
```

```
   [1]_com   [0]_com   [0]_com
   [0]_com   [1]_com   [0]_com
   [0]_com   [0]_com   [1]_com
```

3.8 Операции над матрицами и векторами

Теперь перейдём к описанию функций над интервальными матрицами.

1. Можно заменять элементы в матрице.

```
>>A=infsupdec("[1,2], 2, [3,4]; pi, e, 3");
>>A(2,2)="[2,41]"
A = 2x3 interval matrix
            [1, 2]_com       [2]_com   [3, 4]_com
  [3.1415, 3.1416]_com   [2, 41]_com      [3]_com
```

2. Умножение матриц. Оно возможно, как известно, при согласовании размеров.

```
>>A=infsupdec("1 2 3; 4 5 6");
>>B=infsupdec("1 2;3 4;5 6");
>> A*B
ans = 2x2 interval matrix
   [22]_com   [28]_com
   [49]_com   [64]_com
```

3. Нахождение обратной матрицы (в данном примере можно проследить за изменением декораций).

```
>>A=infsupdec("1 [2,3]; pi, e");
>>B=inv(A)
B = 3x3 interval matrix
[-0.76254, -0.29604]_trv  [0.41254, 0.56104]_trv
[0.34214, 0.88129]_trv   [-0.28053, -0.1089]_trv
```

Отметим, что обращение можно производить только для неособенных матриц.

```
>>A=infsupdec([1 0; 2 0]);
>> inv(A)
ans = 2x2 interval matrix
   [Empty]_trv   [Empty]_trv
   [Empty]_trv   [Empty]_trv
```

3.9 Интервальный метод Ньютона

Интервальный пакет Octave позволяет искать корни функции, используя оператор `fzero`, если ей в качестве аргументов передать рассматриваемую функцию, данный интервал и производную нашей функции. Результатом будут интервалы, содержащие нули этой функции. `fzero` в Octave реализует так называемый интервальный метод Ньютона.

Чтобы понять, каким образом работает интервальный метод Ньютона, надо разобраться, что такое естественное интервальное расширение классического метода.

Естественным интервальным расширением точечного метода называется метод, полученный заменой всех величин в

точечном методе на интервальные и заменой операций на операции интервальной арифметики.

Интервальный метод Ньютона является естественным интервальным расширением классического метода Ньютона для поиска корня на отрезке.

Пусть нам дана дифференцируемая функция f на интервале $\boldsymbol{X}$, у которой есть нуль x^* на этом интервале. Для любой точки $\hat{x} \in \boldsymbol{X}$ верна теорема Лагранжа

$$f(\hat{x}) - f(x^*) = (\hat{x} - x^*) \dots f'(\xi),$$

где ξ — это точка между $\hat{x}$ и x^*. Перепишем эту формулу

$$x^* = \hat{x} - \frac{f(\hat{x})}{f(x_*)}.$$

Пусть $\boldsymbol{f'}(X)$ — это интервальная оценка производной функции $f(x)$ на интервале $\boldsymbol{X}$, тогда $f'(\xi) \in \boldsymbol{f'}(\boldsymbol{X})$. Таким образом, мы можем записать

$$x^* \in \hat{x} - \frac{f(\hat{x})}{\boldsymbol{f'}(\boldsymbol{X})}$$

Итерационный интервальный метод Ньютона, реализованный в интервальном пакете Octave, будет строиться на следующем соотношении

$$\boldsymbol{x}_0 = \boldsymbol{x},$$
$$\boldsymbol{x}_n = \left(\text{mid}\ (\boldsymbol{x}_{n-1}) - \frac{f(\text{mid}\ (\boldsymbol{x}_{n-1}))}{\boldsymbol{f'}(\boldsymbol{x}_{n-1})} \right) \cap \boldsymbol{x}_{n-1}.$$

Рассмотрим пример использования данного метода. Возь-

мём в качестве рассматриваемой функции

$$f(x) = \sqrt{x} + (x+1)\cos(x)$$

на интервале $\boldsymbol{x} = [0, 6]$. Передадим все необходимые аргументы оператору `fzero` и получим следующее.

```
>>f = @(x) sqrt(x)+(x+1).*cos(x);
>>df=@(x) -(x + 1).* sin(x)+cos(x)+1./(2 * sqrt (x));
>>fzero (f, infsup ("[0, 6]"), df)
ans = 2x1 interval vector
        [2.059, 2.0591]
       [4.3107, 4.3108]
```

В первой (также во второй) строке представлена конструкция ядра системы Octave для определения пользовательских функций.

На следующем графике можно увидеть рассматриваемую функцию и визуально убедиться, что нами были получены адекватный результат.

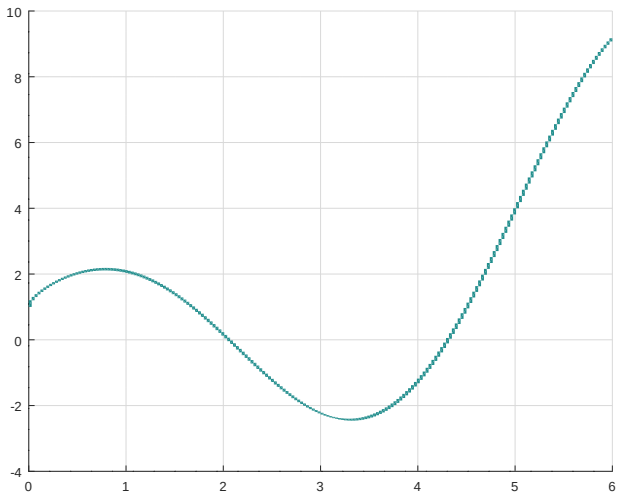

Рис. 3: График функции f

3.10 Множества решений интервальных систем линейных уравнений

Будем рассматривать следующую систему интервальных уравнений

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

где $A = (\boldsymbol{a}_{ij})$, $i, j = 1, \ldots, n$.

Теперь поймём, как можно определить множество решений для ИСЛАУ (интервальных систем линейных алгебраических уравнений). Разберём некоторые множества решений для систем интервальных уравнений.

Начнём с интуитивно понятного определения решения системы. Формальным решением интервального системы уравнений называется интервальный вектор, который после подстановки в систему и выполнению всех операций по правилам интервальной арифметики и анализа обращает систему в систему равенств.

Объединённое множество решений для интервальной системы линейных уравнений $\boldsymbol{A}x = \boldsymbol{b}$ называется

$$\Xi_{uni}(\boldsymbol{A}, \boldsymbol{b}) = \{x \in \mathbb{R}^n \mid (\exists A \in \boldsymbol{A})(\exists b \in \boldsymbol{b})\boldsymbol{A}x = \boldsymbol{b})\}.$$

Допусковым множеством решений для интервальной системы линейных уравнений $\boldsymbol{A}x = \boldsymbol{b}$ называется

$$\Xi_{tol}(\boldsymbol{A}, \boldsymbol{b}) = \{x \in \mathbb{R}^n \mid (\forall A \in \boldsymbol{A})(\exists b \in \boldsymbol{b})(\boldsymbol{A}x = \boldsymbol{b})\}.$$

Управляемым множеством решений для интервальной си-

стемы линейных уравнений $\boldsymbol{A}x = \boldsymbol{b}$ называется

$$\Xi_{ctl}(\boldsymbol{A}, \boldsymbol{b}) = \{x \in \mathbb{R}^n \mid (\forall b \in \boldsymbol{b})(\exists A \in \boldsymbol{A})(Ax = b)\}$$

Большим недостатком интервального пакета является то, что он не имеет встроенных функций для нахождения допускового, объединённого и управляемого множества решений для систем интервальных уравнений, так как они имеют большое значение в интервальном анализе.

Интервальная система может рассматриваться как совокупность классических систем линейных уравнений, которые могут быть решены одновременно. Если к интервальной матрице $\boldsymbol{A}$ и вектору $\boldsymbol{b}$ применить операцию `A\b`, тогда мы получим внешнюю оценку множества решений точечных систем линейных уравнений, то есть внешняя оценка множества $\{x \in \mathbb{R} \mid (\forall A \in \boldsymbol{A})(\forall b \in \boldsymbol{b})(Ax = b)\}$.

Рассмотрим примеры использования данной функции.

```
>> A=infsupdec("1.0001 0.0001; 2.0001 0.0001");
>> b=infsupdec("4.0001; 8.0001");
>> A\b
ans = 2x1 interval vector
     [3.9999, 4.0001]_trv
    [-3.0001, -2.9999]_trv
```

Порассуждаем о том, какие выводы мы можем сделать, посмотрев на решения ИСЛАУ, полученное данной стандартной функцией `A\b`.

В некоторых случаях одна из компонент вектора реше-

ния ИСЛАУ $\boldsymbol{A}x = \boldsymbol{b}$ совпадает со всей вещественной прямой (неограничена), но говорить про то, что линейная система $Ax = b$ ($A \in \boldsymbol{A}$, $b \in \boldsymbol{b}$) не определена, мы не можем. А вот обратное верно.

```
>> A=infsupdec([1 0; 2 0])
>> A\[4;8]
ans = 2x1 interval vector
        [4]_trv
   [Entire]_trv
```

Также бывают случаи, когда хотя бы одна из координат решения ИСЛАУ $\boldsymbol{A}x = \boldsymbol{b}$ пуста, тогда мы можем сказать, что линейная система $Ax = b$ переопределена и не имеет решения. Однако из непустого решения ИСЛАУ мы не можем заключить, имеют ли линейные системы решение или нет.

```
>> A=infsupdec([1 0; 2 0]);
>> A\[3;0]
ans = 2x1 interval vector
   [Empty]_trv
   [Empty]_trv
```

Хочется отметить, что результатом выполнения функции `A\b` не совпадает с результатом `inv(A)*b`. Также решение, полученное применением данной функции, не является формальным.

В следующих разделах сконцентрируемся на объединённом множестве решений и рассмотрим методы для его нахождения в следующих разделах.

3.11 Интервальный метод Гаусса

Классический метод Гаусса является достаточно простым в реализации и понимании методом для решения систем линейных уравнений, основанным на приведении к ступенчатому виду неособенной матрицы правых частей. Он состоит из прямого и обратного хода. Во время первого последовательно зануляются поддиагональные элементы с 1 до $n-1$ столбца, и соответственно меняется вектор правых частей. Происходит приведение матрицы к верхнетреугольному виду. Далее производится обратная подстановка, в которой последовательно происходит вычисляются значения с n-ой неизвестной. Отметим, что если методу где-то встретилось деление на нуль, то метод не может ничего нам сказать о решении.

Перейдём теперь к рассмотрению интервального метода Гаусса и системы интервальных уравнений

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

где $\boldsymbol{A} = (\boldsymbol{a_{ij}})$, $i, j = 1, \ldots, n$.

Интервальный метод Гаусса является естественным интервальным расширением точечного метода Гаусса. Он также состоит из прямого хода и обратной подстановки.

Текст программы, реализующий интервальный метод Гаусса в СКМ Octave, представлен в приложении для ознакомления.

Можно ли сказать, что интервальный метод Гаусса в случае своего успешного завершения (если ему не встретилось

деление на нульсодержащий интервал) выдаёт интервальную оценку для объединённого множества решений? Ответ на этот вопрос положительный в силу основной теоремы интервальной арифметики, применённой к рациональным функциям, которые выдают выражения для компонент вектора решений через элементы $\boldsymbol{A}$ и $\boldsymbol{b}$.

Теперь рассмотрим вопрос о применимости данного метода. Сразу скажем, что он не всегда применим к системам с неособенными матрицами, как его точечный аналог. Рассмотрим так называемый пример Райхмана с матрицей

$$\boldsymbol{A} = \begin{pmatrix} 1 & [0, 0.62] & [0, 0.62] \\ [0, 0.62] & 1 & [0, 0.62] \\ [0, 0.62] & [0, 0.62] & 1 \end{pmatrix}$$

Эта матрица является неособенной. Но при запуске предложенной программы с данной матрицей $\boldsymbol{A}$ и некоторым вектором $\boldsymbol{b}$ мы видим, что метод неприменим.

```
>> aa=[1 0 0;0 1 0;0 0 1];
>> bb=[1 0.62 0.62; 0.62 1 0.62;  0.62 0.62 1];
>> A=infsupdec(aa,bb);
>> b=[infsupdec(1,2);infsupdec(1,2);infsupdec(1,2)];
>>  IGauss(A,b)
ans = 3x1 interval vector
   [Entire]_trv
   [Entire]_trv
   [Entire]_trv
```

Как и в неинтервальном случае мы можем улучшить алго-

ритм выбором ведущего элемента. Это можно осуществить, если стараться максимизировать мигнитуду ведущего элемента.

Но вопрос о необходимом условии применимом интервального метода Гаусса остаётся открытым в данной работе, поскольку требует теоретических выкладок. Ответ на него можно найти в книге [1].

3.12 Интервальный метод Гаусса-Зейделя

Иногда бывают ситуации, в которых нам известно множество внешней оценки решения ИСЛАУ, но оно является очень широким и требует уточнения. В такой ситуации можно применить интервальный метод Гаусса-Зейделя. Также он может быть применён для нахождения внешней оценки части множества решений, ограниченной некоторым брусом.

Снова рассматриваем интервальную систему линейных уравнений $\boldsymbol{A}x = \boldsymbol{b}$ с неособенной интервальной матрицей правых частей. Положим, что в $\boldsymbol{A} = (\boldsymbol{a}_{ij})$ элементы главной диагонали ненулевые. Поскольку матрица $\boldsymbol{A}$ неособенная, этому условию всегда можно удовлетворить перестановкой строк. Пусть известен некоторый брус $\boldsymbol{x}$, содержащий множество решений $\Xi(\boldsymbol{A}, \boldsymbol{b})$ или же некоторую его часть, которая интересует нас по условиям задачи. Тогда можно применить рассматриваемый метод. Опустим его теоретический вывод и представим текст только программы, реализующий его. Его можно найти в приложении.

В реализации интервального метода Гаусса-Зейделя встре-

тится раннее не введённая функция `hdist(x,xx)`. Она позводяет найти расстояние между двумя интервальными объектами и определяется следующим образом

$$\mathrm{hdist}(\boldsymbol{x}, \boldsymbol{y}) = \max\left\{\left|\overline{\boldsymbol{x}} - \overline{\boldsymbol{y}}\right|, \left|\underline{\boldsymbol{x}} - \underline{\boldsymbol{y}}\right|\right\}.$$

Рассмотрим следующую ИСЛАУ.

$$\begin{pmatrix} [5,6] & [-2,-1] \\ [-2,1] & [4,6] \end{pmatrix} x = \begin{pmatrix} 0 \\ 8 \end{pmatrix}$$

Протестируем реализацию метода Гаусса-Зейделя на этом примере.

```
>> x=[5 -2;-2 4];
>> y=[6 -1; 1 6];
>> A=infsupdec(x,y);
>> b=[0;8];
>> x=[infsupdec(-10,10);infsupdec(-10,10)];
>> IGausSeidel(A,b,x)
ans = 2x1 interval vector
   [0.19444, 1.0001]_trv
    [1.1666, 2.5001]_trv
```

3.13 Построение графиков

Octave снабжена аппаратом визуализации, и он позволяет строить как 2D, так и 3D графики и модели.

Для построения двумерных графиков используется функция `plot`. Если ей передать в качестве аргумента интервал $\boldsymbol{y}$,

то на графике будет отображено множество точек $(1, y)$, где $y \in \boldsymbol{y}$.

Для прорисовки двумерного интервала также используется функция `plot`, и ей передаются в качестве значений 2 интервала, из которых составляется брус.

```
>> q=infsupdec(-1,1);
>> plot(q)
```

```
>> q=infsupdec(-1,1);
>> w=infsupdec(3,5);
>> plot(q,w,gray)
```

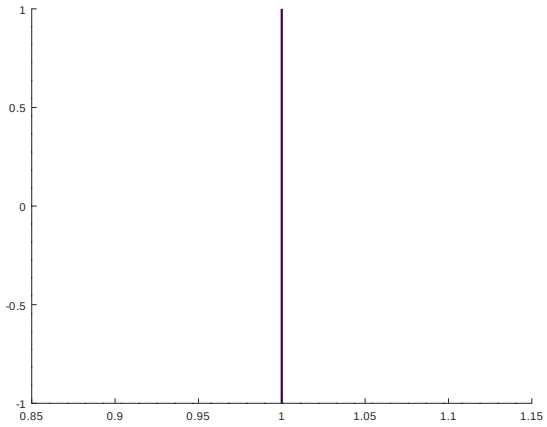

Рис. 4: Одномерный интервал

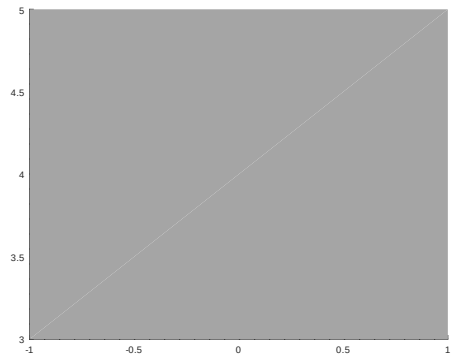

Рис. 5: Двумерный интервал

При необходимости можно менять цвет графика, тип прорисовки линий и другое, добавляя в аргумент некоторые специальные маркеры. Их список можно увидеть либо при вводе `help plot` в командном окне, либо найти в документации. Параметры графика также можно задавать самостоятельно, например, цвет

```
blue = [38 139 210] ./ 255
shade = [238 232 213] ./ 255
gray = [165 165 165] ./255
```

Функция `mince(x,n)` делит рассматриваемый интервал x на заданное число подинтервалов n, которые могут использоваться для оценки диапазона функции. В результате если передать конструктору `plot` в качестве аргуметов рассматриваемую функцию на интервале x и `mince(x,n)`, то на графике мы получим на каждом из подинтервалов внешнюю оценку нашей функции, которая представляет собой прямоугольный брус.

В данном примере мы представили интервальную и классическую прорисовки графика для возможности их сравнения.

```
x = mince (2*infsup (0, "pi"), 20);
plot (x, cos (x), shade)
x = linspace (0, 2*pi, 20);
plot (x, cos (x), 'linewidth', 2, 'color', "k")
```

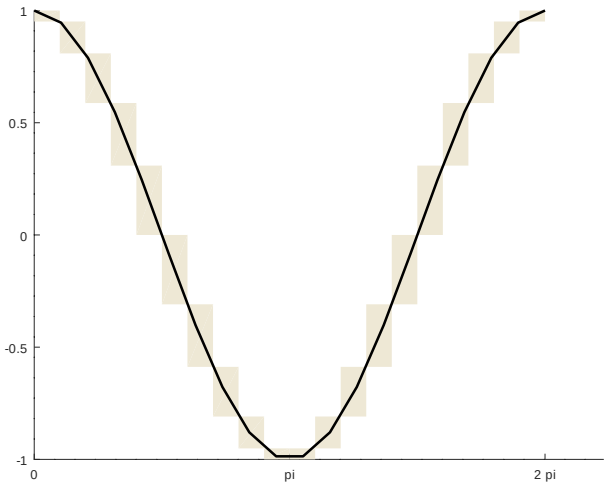

Рис. 6: Интервальная прорисовка графика

Функция `meshgrid(X,Y,Z)` предназначена для создания сетки для трёхмерного графика, которая представляет собой

трёхмерную интервальную матрицу. В качестве аргументов ей передаются вектор-строки или вектор-столбцы, которые в свою очередь преобразуются в матрицы, составленные из исходных повторяющихся строк или столбцов. Если какой-то из аргументов отсутствует, тогда по умолчанию считается, что либо рассматривается двумерная сетка, либо координата Z совпадает с Y.

Приведём интервальное и неинтервальное представление для одной и той же трёхмерной функции.

Рассмотрим на примере функции

```
f=@(x,y)sin(sqrt(x.^2+y.^2))./sqrt(x.^2+y.^2)
```

Справа представлен набор команд для интервальной прорисовки интересующей нас функции, а слева неинтервальной.

```
ran=mince(infsupdec(-8,8),41);
[X,Y]=meshgrid(ran,ran);
Z = f (X, Y);
surf (X, Y, Z);
```

```
ran=linspace(-8,8,41);
[X,Y]=meshgrid(ran,ran);
Z = f (X, Y);
plot3(X, Y, Z);
```

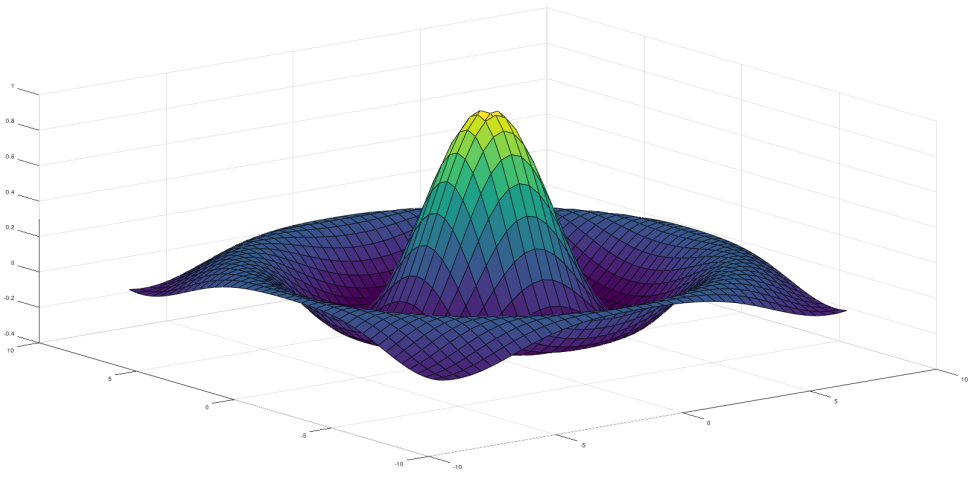

Рис. 7: Классический трёхмерный график

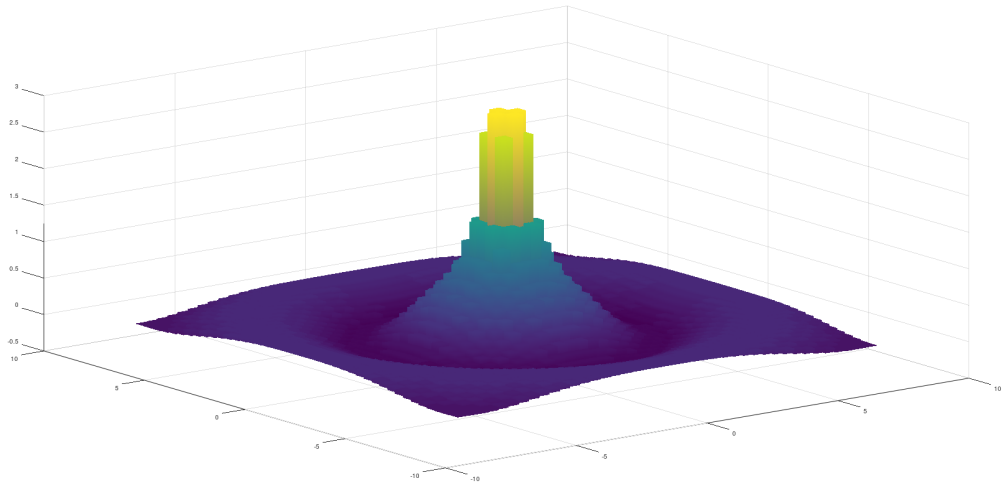

Рис. 8: Интервальный трёхмерный график

Как вы видите трёхмерные графики создаются с помощью функции `plot3`. В качестве аргументов ей передаётся 2 интервала изменения по координатам x и y, сама рассматриваемая функция, а также могут передаваться параметры цвета и прорисовки линий.

4 Заключение

В заключении работы хочется сказать, что интервальный пакет имеет как свои плюсы, так и минусы. Он является свободно распространяемым продуктом, который может помочь исследователю в области интервального анализа получать некоторые практические данные, а также в дальнейшем развивать свои теории. Отметим, что в нём реализованы не все возможности работы с интервальными величинами, но он снабжен основными операциями и функциями, с помощью которых исследователь может полноценно работать.

Также интервальный пакет удволетворяет стандарту IEEE-1788 на интервальные вычисления на ЭВМ, что является боль-

шим плюсом.

В моей работе представлена вводная часть, необходимая для понимания, что такое интервальный анализ. Далее описаны основные операции и функции, реализование в интервальном пакете, правила ввода интервальных величин, рассмотрены некоторые особенности пакета. Также разобраны и реализаны интервальные методы Гаусса, Гаусса-Зейделя и Ньютона для того, чтобы читатель смог увидеть, каким образом описанные в работе теоретические знания можно применять для конкретных задач.

5 Приложение

5.1 Приложение А

Реализация интервального метода Гаусса в СКМ Octave.

```
function x = IGauss(A,b)
%  проверяем корректность входных данных
  m = size(A,1);
  n = size(A,2);
  if m!=n
      error('Матрица системы не квадратная')
  endif
  l = size(b,1);
  if l!=n
     error('Размерность вектора и матрицы не равны')
  endif
%прямой ход метода Гаусса
  for j=1:n-1
    for i=j+1:n
      r(i,j)=A(i,j)/A(j,j);
      for k=j+1:n
        A(i,k)=A(i,k)-r(i,j)*A(j,k);
      endfor
      b(i)=b(i)-r(i,j)*b(j);
    endfor
  endfor
  x=infsupdec(zeros(n,1));
%обратный ход метода Гаусса
```

```
  for i=n:(-1):1
    summ=0;
    for j=i:n
      summ=summ+A(i,j)*x(j);
    endfor
    x(i)=(b(i)-summ)/A(i,i);
  endfor
endfunction
```

5.2 Приложение Б

Реализация интервального метода Гаусса-Зейделя в СКМ Octave.

```
function x = IGausSeidel(A,b,x)
%  проверка на корректность входных данных
  m = size(A,1);
  n = size(A,2);
  if ( m!=n ) then
      error('Матрица не квадратная')
  endif
  mb = size(b,1);
  nb = size(b,2);
  if ( mb!=m | nb>1 ) then
      error('Несоответствие размеров матр. и в-ра')
  endif

%Реализация метода
  Eps = 1.e-5;%допустимая погрешность
  xx = x;
  q=1000;
  while (q>Eps)
    for i=1:n
      Summ=0;
      summ=0;
      for j=1:(i-1)
        summ=summ+A(i,j)*xx(j);
```

```
      endfor
      for j=i+1:n
        Summ=Summ+A(i,j)*x(j);
      endfor
      xx(i)=intersect(x(i),(b(i)-summ-Summ)/A(i,i));
      if isempty(xx(i))
        error('множество решений не пересекает брус')
      endif
    endfor
    q=hdist(x,xx);
    disp(q);
    x=xx;
  endwhile
endfunction
```

5.3 Приложение B

Ниже представлены функции, предназначенные для работы с интервальным пакетом Octave, а также их краткое описание (часть других операторов была рассмотрена в тексте работы). Здесь представлены самые часто встречаемые, остальные можно найти в документации.

Унарные функции.

Название	Выходные данные
`inf(x)`	Левая граница интервала $\boldsymbol{x}$
`sup(x)`	Правая граница интервала $\boldsymbol{x}$
`mid(x)`	Середина интервала $\boldsymbol{x}$
`wid(x)`	Ширина интервала $\boldsymbol{x}$
`rad(x)`	Радиус интервала $\boldsymbol{x}$
`mag(x)`	Магнитуда интервала $\boldsymbol{x}$
`mig(x)`	Мигнитуда интервала $\boldsymbol{x}$
`abs(x)`	Выдаёт неотрицательную часть интервала (модуль интервала)
`floor(x)`	Округляет границы интервала в направлении $-\infty$ до целого числа
`ceil(x)`	Округляет границы интервала в направлении $+\infty$ до целого числа

Интервальные функции.

Название	Выходные данные
`realsqrt(x)`	$\sqrt{x}$ для всех $x \in \boldsymbol{x}$, $x \geq 0$
`pown(x,n)`	x^n для всех $x \in \boldsymbol{x}$
`pow(x,y)`	x^y для всех $x \in \boldsymbol{x}$, $y \in \boldsymbol{y}$, $x \geq 0$, $y > 0$
`exp(x)`	e^x для всех $x \in \boldsymbol{x}$
`pow2(x)`	2^x для всех $x \in \boldsymbol{x}$
`log(x)`	$ln(x)$ для всех $x \in \boldsymbol{x}$
`log2(x)`	$log_2(x)$ для всех $x \in \boldsymbol{x}$
`log10(x)`	$log_{10}(x)$ для всех $x \in \boldsymbol{x}$
`sin(x)`	$sin(x)$ для всех $x \in \boldsymbol{x}$
`cos(x)`	$cos(x)$ для всех $x \in \boldsymbol{x}$
`tan(x)`	$tg(x)$ для всех $x \in \boldsymbol{x}$
`asin(x)`	$arcsin(x)$ для всех $x \in \boldsymbol{x}$, $x \in [-1, 1]$
`acos(x)`	$arccos(x)$ для всех $x \in \boldsymbol{x}$, $x \in [-1, 1]$
`atan(x)`	$arctg(x)$ для всех $x \in \boldsymbol{x}$
`sinh(x)`	$sh(x)$ для всех $x \in \boldsymbol{x}$
`cosh(x)`	$ch(x)$ для всех $x \in \boldsymbol{x}$
`tanh(x)`	$th(x)$ для всех $x \in \boldsymbol{x}$
`asinh(x)`	$arsh(x)$ для всех $x \in \boldsymbol{x}$
`acosh(x)`	$arch(x)$ для всех $x \in \boldsymbol{x}$
`atanh(x)`	$arth(x)$ для всех $x \in \boldsymbol{x}$

Бинарные функции.

Название	Выходные данные
`intersect(x,y)`	Пересечение интервалов $\boldsymbol{x}$ и $\boldsymbol{y}$.
`union(x,y)`	Интервал, который содержит интервалы $\boldsymbol{x}$ и $\boldsymbol{y}$
`min(x,y)`	Вычисляется по формуле $\min(\boldsymbol{x},\boldsymbol{y}) = ((\boldsymbol{x}+\boldsymbol{y}) - \mathrm{abs}(\boldsymbol{x}-\boldsymbol{y}))/2$
`max(x,y)`	Вычисляется по формуле $\min(\boldsymbol{x},\boldsymbol{y}) = ((\boldsymbol{x}+\boldsymbol{y}) + \mathrm{abs}(\boldsymbol{x}-\boldsymbol{y}))/2$

Булевские функции.

Название	Выдаёт истину (1), когда
`isempty(x)`	$\boldsymbol{x}$ является пустым
`isentire(x)`	$\boldsymbol{x}$ совпадает с $\mathbb{R}$
`eq(x,y)`	$\boldsymbol{x}$ и $\boldsymbol{y}$ совпадают
`subset(x,y)`	$\boldsymbol{x}$ является подмножеством $\boldsymbol{y}$
`le(x,y)`	$\boldsymbol{x}$ меньше или равен $\boldsymbol{y}$
`precedes(x,y)`	$\boldsymbol{x}$ меньше $\boldsymbol{y}$, интервалы могут иметь точку пересечения
`interior(x,y)`	Каждая точка интервала $\boldsymbol{x}$ содержится в $\boldsymbol{y}$, но не является границей $\boldsymbol{y}$
`lt(x,y)`	$\boldsymbol{x}$ строго меньше $\boldsymbol{y}$
`strictprecedes`	$\boldsymbol{x}$ находится строго слева от $\boldsymbol{y}$
`disjoint(x,y)`	$\boldsymbol{x}$ и $\boldsymbol{y}$ не пересекаются
`iscommoninterval(x)`	$\boldsymbol{x}$ является ограниченным и непустым
`ismember(a,x)`	Содержит $\boldsymbol{x}$ точку a

Функции, связанные с декорациями.

Название	Описание
`newdec(x)`	Даёт интервалу подходящую декорацию
`intervalpart(x)`	Возвращает недекорированный интервал
`decorationpart(x)`	Выдаёт декорацию интервала

Список литературы

[1] С.П. Шарый Конечномерный интервальный анализ. – Новосибирск: XYZ, 2018.–623 с.

[2] IEEE Standard for Interval Arithmetic: IEEE Std 1788TM-2015.

[3] https://octave.sourceforge.io/interval/package_doc/index.html#SEC_Contents

[4] Введение в Octave для инженеров и математиков:/ Е. Р. Алексеев, О. В. Чеснокова — М.: ALT Linux, 2012. — 368 с.

[5] Interval Analysis in MATLAB, G. I. Hargreaves, Numerical Analysis Report No. 416, December 2002

[6] W.M. Kahana A more complete interval arithmetic. — Departments of Mathematics and of Computer Science, University of Toronto, 1968 — 47 с.

[7] Ю.И. Шокин Интервальный анализ.— Новосибирск: Наука, 1981. — 111 с.

[8] E. Kaucher Interval Analysis in the Extended Interval Space $\mathbb{IR}$. — Computing, Suppl. 2. 33 - 49 (1980), by Springer-Verlag, 1980 — 49 с.

[9] Introduction to interval analysis:/ R.E. Moore, R.B. Kearfott, M.J. Cloud — Philadelphia: SIAM, 2009 — 223 с.