МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Алтайский государственный университет»

Факультет математики и информационных технологий
Кафедра информатики

РАЗРАБОТКА ПАКЕТА ФУНКЦИОНАЛЬНЫХ УЗЛОВ СИСТЕМЫ KNIME ДЛЯ ИНТЕРВАЛЬНОГО АНАЛИЗА ДАННЫХ

Выпускная квалификационная работа магистра

Выполнил студент
2 курса, 447м-ВМ группы
Шипилов В.С.
_____
(подпись)
Научный руководитель
к.ф.- м.н., доцент
Жилин С.И.
_____
(подпись)

Допустить к защите
Зав. кафедрой
к.ф.- м.н.
Жариков А. В.
_____
(подпись)
«___»_____2016 г.

Выпускная квалификационная работа защищена
«___»_____2016 г.

Оценка_____
Председатель ГЭК
_____
_____
(подпись)

Барнаул 2016

РЕФЕРАТ

Выпускная квалификационная работа магистра: 52 страницы, 18 рисунков, три листинга кода, две таблицы, приложение, 30 слайдов, 39 источников литературы.

Объект исследования: методы интервального анализа данных и их практическое применение.

Цель работы: адаптация существующих модулей расширения аналитической платформы KNIME, реализующих методы построения и анализа интервальной регрессии, и разработка модуля, для визуализации множеств решений интервальных систем линейных алгебраических уравнений.

Задачи:

- Проанализировать существующие функциональные узлы, входящие в пакет KNIME Interval Tools, и провести доработку, позволяющую использование узлов на актуальной версии платформы KNIME;
- Разработать модуль расширения системы KNIME, позволяющий визуализировать множество решений интервальных систем линейных алгебраических уравнений;
- Провести опытную эксплуатацию разработанных и адаптированных программных инструментов, в том числе, при решении прикладных задач обработки данных с интервальной неопределенностью.

В выпускной работе проведен анализ существующих функциональных узлов для интервального анализа данных в системе KNIME, который выявил необходимость их адаптации под текущие обновления системы KNIME и библиотек, которые эти узлы используют. Проведена адаптация узлов, входящих в пакет KNIME Interval Tools. Разработан самостоятельный узел, который позволяет визуализировать множество решений интервальных систем линейных алгебраических уравнений. Готовые узлы могут использоваться для решения прикладных и учебных задач интервального анализа данных.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность. В подавляющем большинстве прикладных задач исследователь имеет дело с неточными исходными данными, неопределенность которых порождается различными факторами. В зависимости от источника неточности и неопределенности данных в настоящее время используются различные модели описания неопределенных данных, включая вероятностную, нечеткую и интервальную модели. Каждая из этих моделей имеет свою парадигму, опирается на соответствующий теоретический аппарат, имеет свои методы анализа и область применения. Ряд работ А.Ф. Бочкова, А.П. Вощинина, Л.В. Канторовича, А.В. Максимова, Н.М. Оскорбина, С.И. Жилина и других показывают, что применение именно интервального анализа к широкому спектру прикладных задач с ограниченными ошибками и неопределенностью в данных имеет большие перспективы.

Применение интервального анализа позволяет снять многие проблемы и методические сложности, возникающие при решении прикладных задач статистическими методами. В рамках интервального анализа неопределенность исходных данных может иметь разные источники и природу. Интервал неопределенности позволяет описать широкий класс неопределенных, неоднозначных, вариабельных и неточных исходных данных. Значения ошибок в исходных данных могут колебаться в широких пределах. Результаты, полученные с помощью парадигмы интервального анализа, имеют ясную и четкую интерпретацию в терминах интервалов и областей неопределенности.

Набор интервальных программных средств довольно широк, однако, у каждого типа программных средств есть своя специфика, свои библиотеки и прочее. В задачах анализа данных наиболее целесообразно использовать специальные платформы, которые сочетают в себе разнородные инструменты упрощающие программирование за счет использования графического интерфейса, различных сценариев и т.п.

На сегодняшний момент существует немало систем анализа данных или аналитических платформ (KNIME, RapidMiner и др.). В составе платформы KNIME реализованы инструменты для интервального анализа данных в рамках проекта KNIME Interval Tools. Однако эти инструменты требуют адаптации под нынешние обновления системы KNIME, есть потребность пополнения новыми инструментами, в частности для визуализации.

Целью настоящей работы является адаптация существующих модулей расширения аналитической платформы KNIME, реализующих методы построения и анализа интервальной регрессии, и разработка модуля, для визуализации множеств решений интервальных систем линейных алгебраических уравнений.

Для реализации поставленной цели необходимо решить следующие **задачи:**

- Проанализировать существующие функциональные узлы, входящие в пакет KNIME Interval Tools, и провести доработку, позволяющую использование узлов на актуальной версии платформы KNIME;
- Разработать модуль расширения системы KNIME, позволяющий визуализировать множество решений интервальных систем линейных алгебраических уравнений;
- Провести опытную эксплуатацию разработанных и адаптированных программных инструментов, в том числе, при решении прикладных задач обработки данных с интервальной неопределенностью.

Объектом исследования работы являются методы интервального анализа данных и их практическое применение.

Предметом исследования данной работы является программная реализация функциональных узлов в системе KNIME, позволяющих использовать методы интервального анализа в сценариях анализа данных.

Практическая значимость: функциональные узлы могут использоваться в практических и учебных задачах интервального анализа данных

Апробация работы. Основные положения и отдельные результаты работы докладывались и обсуждались на Всероссийской конференции “Математика и её приложения: фундаментальные проблемы науки и техники” (Барнаул, 2015), Третьей Региональной конференции “Мой выбор – НАУКА!” (Барнаул, 2016).

Публикации. По теме диссертации опубликована работа [39].

Структура и объём работы. Выпускная квалификационная работа состоит из введения, трех глав, заключения, 18 рисунков, списка литературы из 39 источников, приложения. Полный объём 52 страницы.

В первой главе рассматриваются основные понятия, методы и программные средства интервального анализа данных. Вторая глава посвящена разработке функциональных расширений системы KNIME для интервального анализа данных. В третьей главе приведены краткое руководство пользователя и примеры решения задач анализа данных с использованием разработанных модулей расширения KNIME.

ГЛАВА 1. ИНТЕРВАЛЬНЫЙ АНАЛИЗ ДАННЫХ: МЕТОДЫ И ПРОГРАММНЫЕ СРЕДСТВА

1.1. Интервальный анализ и его применение

В этом разделе приводятся основные сведения о постановках задач и основные методы интервального анализа данных. Изложенные ниже сведения частично заимствованы из [1].

В практических задачах анализа данных, как правило, приходится учитывать неопределенности разного рода, связанные, например, с неточностью измерений величин, с неполнотой знаний о моделируемом объекте и т.п. Одним из наиболее естественных способов описания неопределенностей, является их интервальное представление. Далее интервалом $\boldsymbol{x} = [\underline{x}, \overline{x}]$ называется множество всех действительных чисел, расположенных между заданными граничными числами $\underline{x}$ и $\overline{x}$, включая их самих, т.е. $\boldsymbol{x} = \{x \in \mathbb{R} \mid \underline{x} \leq x \leq \overline{x}\}$. Интервальная неопределённость – это состояние неполного (частичного) знания об интересующей нас величине, когда известна лишь её принадлежность некоторому интервалу, т. е. когда мы можем указать лишь границы возможных значений этой величины (пределы её изменения). Соответственно, интервальный анализ – это раздел математики, предметом которого является решение задач с интервальными (или, более общо, ограниченными) неопределённостями и неоднозначностями в данных, возникающими либо в условиях задачи, либо в процессе её решения, чьей характеристической особенностью является рассмотрение множеств неопределённости как самостоятельных целостных объектов, посредством установления арифметических, аналитических и т. п. операций и отношений между ними. Интервальный анализ и его специфичные методы имеют, таким образом, наивысшую ценность в задачах, где неопределённости и неоднозначности возникают с самого начала и являются неотъемлемой частью

постановки задачи. Особого внимания в интервальном анализе данных требуют: интервальные системы линейных алгебраических уравнений, интервальные методы глобальной оптимизации, построение зависимостей одного признака от совокупности факторов и др. Далее для настоящей работы востребованными являются интервальные системы линейных алгебраических уравнений (ИСЛАУ), т.е. системы вида: $\boldsymbol{Ax} = \boldsymbol{b}$, где $\boldsymbol{A}$ – интервальная $(m \times n)$ – матрица, а $\boldsymbol{b}$ – интервальный вектор-столбец размера m. Матрица или вектор называются интервальными, если их элементами являются интервалы. Понятие множества решений ИСЛАУ может наполняться различным смыслом в зависимости от постановки прикладной задачи, сводящейся к ИСЛАУ, и способа интерпретации интервалов (подробнее см. в [1]). Наиболее часто речь идет о таких множествах решений, как

- объединенное:
  $\Xi_{uni}(\boldsymbol{A}, \boldsymbol{b}) = \{x \in \mathbb{R}^n \mid (\exists A \in \boldsymbol{A})(\exists b \in \boldsymbol{b})(Ax = b)\}$;
- допусковое:
  $\Xi_{tol}(\boldsymbol{A}, \boldsymbol{b}) = \{x \in \mathbb{R}^n \mid (\forall A \in \boldsymbol{A})(\exists b \in \boldsymbol{b})(Ax = b)\}$;
- управляемое:
  $\Xi_{ctl}(\boldsymbol{A}, \boldsymbol{b}) = \{x \in \mathbb{R}^n \mid (\forall b \in \boldsymbol{b})(\exists A \in \boldsymbol{A})(Ax = b)\}$.

В прикладных задачах чаще всего оперируют внутренними и внешними оценками этих множеств. Вне зависимости от типа множества решений ИСЛАУ представляют собой политоп – объединение конечного числа полиэдров. Полиэдром в $\mathbb{R}^n$ называют подмножество пространства $\mathbb{R}^n$, которое представимо, как множество решений системы линейных неравенств вида $Ax \geq b$ относительно неизвестного x, в которой $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $m, n \in \mathbb{N}$. Предполагается, что элементы пространства $\mathbb{R}^n$ – это вектор-столбцы.

Строение множества решений ИСЛАУ может быть довольно сложным и в общем случае построение его точного описания представляет собой процедуру, количество операций в которой экспоненциально зависит от

размерности пространства [2]. Универсальный метод решения задачи распознавания разрешимости может быть основан на том факте, что пересечения множества решений ИСЛАУ с каждым из ортантов (координатных углов) пространства $\mathbb{R}^n$ являются выпуклыми полиэдральными множествами, для которых уравнения граничных гиперплоскостей легко выписываются по матрице и правой части ИСЛАУ. Как следствие, пустота или непустота пересечения $\Xi(\boldsymbol{A}, \boldsymbol{b})$ с каждым из ортантов $\mathbb{R}^n$ может быть выявлена путём решения некоторой системы линейных неравенств, например, с помощью хорошо разработанных методов линейного программирования. Кроме того, полное описание множества решений трудно для восприятия лицом, принимающим решения. По этим причинам на практике часто довольствуются теми или иными оценками множеств решений ИСЛАУ. Тем не менее, в низкоразмерных случаях при работе с ИСЛАУ бывает полезно визуализировать множество ее решений, его проекций или сечений, чтобы упростить их анализ и тактику оценивания. Визуализация политопов, представляющих множество решений ИСЛАУ, может осложняться в связи с тем, что в зависимости от коэффициентов в левых и правых частях ИСЛАУ, множество решений может быть пустым, неограниченным и/или невыпуклым.

Не менее актуальной постановкой задачи интервального анализа данных является интервальная регрессия – метод линейного моделирования и построения интервальных оценок прогноза для многомерных данных с ограниченной ошибкой отклика [3]. Основным предположением интервальной регрессии является ограниченность погрешности отклика, в отличие от большинства статистических методов, которые предполагают нормальность погрешности. Если положить, что при точном измерении входной переменной отклик измерен с погрешностью ε то:

$$y_p \in [y - \varepsilon, y + \varepsilon], \tag{1.1}$$

где y_p – точный отклик, y – измеренный отклик, ε – погрешность измерения. Часто ограничения погрешности могут быть добыты из технического паспорта

прибора, которым производятся измерения, либо можно оценить погрешность из собственных соображений (основываясь на знании метода получения данных). В настоящей работе модель регрессии предполагается линейной:

$$x = (1\ x_1\ x_2\ \dots\ x_n),$$
$$a = (a_0\ a_1\ \dots\ a_n), \qquad (1.2)$$
$$y = ax,$$

где a – вектор параметров, x – вектор независимых переменных, y – отклик. Решением регрессии является множество допустимых значений вектора параметров модели a:

$$A = \{a |\ y_i - \ \varepsilon_i\ \le ax^i \le y_i + \ \varepsilon_i, i = 1..n\}. \qquad (1.3)$$

Любой вектор $a \in A$ может быть выбран для построения модели линейной регрессии (1.2). В связи с этим построенное множество называют множеством неопределённости параметров, что совпадает с терминологией работы [4]. Каждая точка множества A задаёт некоторую модель зависимости, а в совокупности эти модели образуют множество допустимых моделей, которое в некоторых случаях может представлять собой самостоятельный предмет интереса. Множество A является выпуклым политопом. Часто для анализа множества неопределённости выбирают более простое множество (многомерный параллелепипед с гранями параллельными координатным плоскостям), которое содержит множество A. Это связано с большой сложностью работы с произвольным многогранником высокой размерности. Самый маленький из политопов, содержащих множество неопределённости находится решением задач линейного программирования:

$$\underline{a_i} = \min_{a \in A} a_i,$$
$$\overline{a}_i = \max_{a \in A} a_i.$$

Множество неопределённости (1.3) дает возможность построить интервальную оценку отклика $y(x) \in \left[\underline{y}(x), \overline{y}(x)\right]$, где

$$\underline{y}(x) = \min_{a \in A} ax,$$

$$\overline{y}(x) = \max_{a \in A} ax.$$

В качестве точечной оценки и разброса могут быть выбраны $\hat{y}$, которые совпадают в работах [3-5].

$$\widehat{y_i} = \frac{\underline{y}(x) + \overline{y}(x)}{2},$$

$$\Delta y = \frac{\overline{y}(x) - \underline{y}(x)}{2}.$$

В анализе данных ошибочные наблюдения называют выбросами – это образцы, выбивающиеся из общей совокупности и возникающие под действием случайных или намеренных помех. Попадание выбросов в выборочные данные может быть обусловлено рядом причин: неисправностью оборудования, с которого снимались показатели, необычными помехами при сборе данных или, что чаще всего, ошибками оператора при вводе информации. В случае возникновения выбросов в методе регрессии для интервальных данных происходит разрушение модели при добавлении очередного наблюдения. Однако не только выбросы мешают построению модели, но и недооценённость погрешности. Такая ситуация возникает чаще всего, если предельный уровень погрешности носит характер априорного предположения. Один из методов выявления выбросов основан на поиске наименьшего допустимого уровня погрешности, позволяющего получить непустое множество неопределенности. В конечном итоге оценивать погрешность или не принимать определённый объект в рассмотрения, считая его выбросом, определяется только условиями прикладной задачи.

Задача регрессионного анализа состоит в построении эмпирической зависимости между откликом и значением параметра. Задача сводится к определению параметров модели из эмпирических наблюдений, определения интервала значений отклика при любом значении параметра, построения множества неопределённости, точечного значения отклика параметра. Используемый метод постулирует некоторые факты: ограниченность ошибки отклика (1.1), априорная определённость вида взаимосвязи (1.2) и адекватность

любой модели значения параметров, которой находится во множестве (1.3). Цель построения регрессии – определение значения отклика по значению признака. Из множества неопределённости после построения зависимости по методу интервальной регрессии можно сделать заключения о значении отклика для любого признака. В случае если данные непротиворечивы, можно определить все интересующие величины как решения соответствующих задач линейного программирования. Решение таких задач чаще всего сопряжено с полным перебором вариантов, для ускорения процесса решения применяют симплекс метод [6]. На практике, много задач сводится к решению задач интервального анализа данных. Далее приведены примеры задач с интервальной неопределенностью.

Геометрические преобразования и привязка изображений. Ярким примером может служить использование данных дистанционного зондирования в геоинформационных технологиях. В частности, при решении задач мониторинга экосистем необходимо сопоставлять спутниковые снимки одной и той же территории, сделанные в разное время, и осуществлять их координатную привязку с одновременной геометрической коррекцией для последующего совместного анализа. Для построения геометрического преобразования изображения требуется установить соответствие между элементами преобразуемого и эталонного изображения, что сводится к выделению так называемых сопряженных (реперных, контрольных, опорных) точек на изображениях. Точки на двух изображениях называются сопряженными, если они являются образами одной и той же точки сцены. Для решения этой задачи наиболее преимущественным является метод центра неопределенности [7]. Разработка методов анализа пространственной неопределенности при выполнении любых операций над пространственными данными является одним из приоритетных направлений геоинформационных технологий [8-12].

Распознавание изображений. Одна из востребованных на практике постановок задач анализа изображений заключается в классификации

изображений, на которых запечатлены некоторые эталонные образы, съемка которых искажена действием каких-либо помех. Типичным примером такой ситуации является автоматическое распознавание цифр на автомобильных номерах. Эталонные изображения символов, используемых на автомобильных номерах, заранее известны, но на изображениях, получаемых камерами видеонаблюдения, символы могут быть загрязнены или содержать блики. Решается задача алгоритмом, который предложен в [13, 14], а частности, для реализации алгоритма необходимо отыскать объединенное множество решений системы линейных алгебраических уравнений, подробнее в [1].

Анализ электрических цепей. Задача анализа линейной электрической цепи заключается в определении характеристик элементов цепи, при которых будет осуществляться ее стабильное функционирование. Структура электрической цепи накладывает ограничения на распределение токов и напряжений на отдельных ее элементах. Имея цепь заданной структуры, а также интервальные допуски на значения а) сопротивления на ветвях цепи и б) напряжения на источниках тока, требуется определить допуски на значения в) силы тока в ветвях цепи и г) напряжения в узлах цепи. Один из способов решения интервальной задачи о допусках электрической цепи состоит в ее сведении к задаче оценивания допускового множества решений интервальной системы линейных алгебраических уравнений [15]. Подробное обсуждение постановки линейной интервальной задачи о допусках, а также методов, позволяющих установить ее разрешимость и отыскать внутреннюю оценку множества $\Xi_{tol}(\boldsymbol{A}, \boldsymbol{b})$ в виде интервального вектора, можно обнаружить в [1]. Один из простых способов построения внутренней оценки допускового множества решений дает метод Шайдурова [16, 17]. Идея этого метода заключается в поиске точки допускового множества решений и «раздутии» интервального бруса, имеющего центром эту точку и целиком лежащего в допусковом множестве решений.

1.2. Интервальные программные средства

Список существующего интервального программного обеспечения [18] довольно широк, неуклонно пополняется и включает программные продукты разного уровня и назначения: от простейших интервальных калькуляторов до специализированных интервальных языков программирования. Значимое место в этом ряду занимают библиотеки интервальных подпрограмм для универсальных языков программирования, поскольку служат инструментарием разработки не только систем научных вычислений, но и прикладного программного обеспечения различного назначения. Интервальные библиотеки для таких традиционных языков научных вычислений, как `C/C++` и `Fortran`, весьма многочисленны, а некоторые из них даже претендуют на статус стандартных компонент языка. Относительно недавно и на языке Java был запущен и до сих пор развивается проект разработки открытой интервальной `Java`-библиотеки `JInterval`. Основными принципами проекта являются:

- открытость проекта;
- обеспечение гибкости в выборе приоритетов вычислений – высокая скорость или высокая точность;
- расширенная функциональность.

Структурно библиотека `JInterval` разделена на четыре функциональных слоя:

- элементарные интервальные арифметические операции;
- элементарные интервальные функции;
- интервальные векторы, матрицы и операции над ними;
- высокоуровневые интервальные методы.

В рамках первого функционального слоя библиотеки служит слой интервальных арифметик (классической, расширенной классической, Каухера), реализованных в соответствии с проектом стандарта интервальной арифметики IEEE 1788 [19]. Реализации интервальных арифметик допускают гибкое управление представлением границ интервалов (в виде рациональных чисел с

абсолютной точностью, а также в форме чисел с плавающей точкой различной длины) и режимами округления при вычислениях. Поверх каждой из арифметик надстроен слой, включающий в себя интервальные элементарные арифметические, тригонометрические и теоретико-множественные функции. На третьем уровне реализована поддержка интервальных векторов и матриц и основные операции над ними. В четвертом слое доступны методы оценивания множеств решений интервальных систем линейных алгебраических уравнений, а также средства построения и анализа регрессионных зависимостей по интервальным данным и многое другое. Исходные коды библиотеки свободно доступны в Интернет [20].

В практике анализа данных широко используются такие специализированные платформы, как KNIME [21], RapidMiner [22] и им подобные. Помимо дружественного интерфейса, функционального богатства и гибкости одним из достоинств этих систем, является их платформонезависимость, обеспечивающаяся использованием Java-технологий. Функциональность этих систем может быть пополнена сторонними расширениями, разработанными на Java).

Система KNIME представляет собою модульную платформу с открытыми исходными кодами, предназначенную для анализа данных. Основными элементами в этой системе являются узлы – готовые инструменты позволяющие делать различные операции с данными. KNIME позволяет визуально конструировать потоковые сценарии обработки и анализа данных из стандартных функциональных узлов, таких как чтение данных, предобработка данных, обучение модели, применение модели, визуализация данных и т.п. Сценарии могут быть исполнены частично или целиком, а промежуточные или окончательные результаты обработки и анализа данных могут быть изучены с использованием разнообразных инструментов визуализации рис. 1.1.

Рисунок 1.1. Пример сценария в системе KNIME

Основные преимущества системы KNIME перед схожими системами:

- кроссплатформенность;
- открытый исходный код платформы;
- открытый исходный код всех узлов;
- бесплатная платформа;
- более 100 узлов для разных типов анализа данных;
- возможность установки сторонних (пользовательских) модулей;
- простота разработки узлов;
- большое сообщество разработчиков.

К сожалению, в платформе KNIME отсутствуют интервальные инструменты, которые можно было бы гибко комбинировать с прочими инструментами анализа данных. Тем не менее, платформа KNIME предоставляет весьма богатый фундамент для разработки интервальных инструментов.

Создание расширений в системе KNIME, реализующих методы интервального анализа началось разработкой набора узлов, позволяющих решать задачи построения и анализа регрессионных зависимостей при интервальной ошибке в откликовой переменной [23]. Со временем стали

появляться и другие узлы. На текущий момент в Алтайском государственном университете созданы следующие функциональные узлы для интервального анализа в системе KNIME в рамках проекта KNIME Interval Tools [24-26]:

- Interval Regression (Learner) – узел построение модели интервальной регрессии, на вход приходит таблица данных $(X, y, \vec{\varepsilon})$, а на выходе получается таблица $(X, y, \bar{\varepsilon}, \hat{\boldsymbol{y}})$, где $\hat{\boldsymbol{y}} = \left[\underline{y}, \overline{y}\right]$ – интервальные прогнозы в точках X (коридор допустимых моделей);
- Interval Regression (Predictor) – узел построение прогноза по заданным значениям независимой переменной и модели интервальной регрессии. На вход в этот узел поступает построенная в узле Interval Regression (Learner) модель (верхний вход) в виде $(X, y, \vec{\varepsilon})$ и поток данных, содержащих значения независимых переменных X' (нижний вход), а на выходе получается таблица данных $(X', \hat{\boldsymbol{y}}')$, где $\hat{\boldsymbol{y}} = \left[\underline{y}', \overline{y}'\right]$ – интервальные прогнозы в точках X';
- IR Consistency – узел проверяет, является ли данные для интервала регрессии совместимыми. Входная таблица $(X, y, \vec{\varepsilon})$, выход: ИСТИНА, если данные и структура совместны, ЛОЖЬ – иначе;
- IR Outlier Detector – узел, определяющий коэффициенты растяжения интервалов ошибки, в него приходят обучающие данные $(X, y, \vec{\varepsilon})$, а на выход выдаются данные в виде $(X, y, \bar{\varepsilon}, \omega)$, где ω – столбец коэффициентов расширения интервальных наблюдений, необходимого для достижения совместности данных и структуры модели;
- ILS Solver – решатель интервальных систем линейных алгебраических уравнений $\boldsymbol{A}x = \boldsymbol{b}$, на вход приходят данные $(\boldsymbol{A}, \boldsymbol{b})$, на выходе получается таблица x, где $x = \left[\underline{x}, \overline{x}\right]$ – оценка решения. Модуль позволяет получать три вида решений ИСЛАУ $\boldsymbol{A}x = \boldsymbol{b}$, а именно: формальное решение в полной арифметике Каухера, внутреннюю и внешнюю оценки объединенного множества решений.

Приведенные выше функциональные узлы используют библиотеку JInterval, однако, ее не достаточно для их функционирования. Узел ILS Solver для оценки решений использует узкоспециализированную пользовательскую библиотеку, которая решает задачи линейного программирования симплекс-методом – lpsolve [27]. Библиотеки, которые необходимы для работы функциональных узлов, стремительно улучшаются, расширяются и переписываются, к сожалению, проект KNIME Interval Tools не успевает развиваться с той же скорость. По этой причине актуальной представляется задача адаптировать вышеуказанные узлы под текущие обновления библиотек и обновления самой платформы KNIME. Дополнительно есть потребность в пополнении пакета KNIME Interval Tools узлом, который будет визуализировать множества решений интервальных систем линейных алгебраических уравнений. Проблема визуализации множеств решений ИСЛАУ не нова. Так в [28] В. Крамером предложен метод визуализации, который реализован его учеником Г. Повом с использованием Java3D. Однако этот продукт известен лишь в узких кругах, и не всегда обеспечивает корректные решения в "сложных" случаях. Подход, охватывающий все особенные ситуации при визуализации множеств решений ИСЛАУ, предложен И.А. Шарой [29] и получил название "метод граничных интервалов". Имеется авторская реализация метода граничных интервалов в виде пакетов MATLAB [30].

1.3. Метод граничных интервалов для визуализации множеств решений интервальных систем линейных алгебраический уравнений

Метод граничных интервалов – это метод исследования и визуализации полиэдральных множеств в евклидовых пространствах. Основная идея метода для простоты здесь кратко излагается в случае. Граничный интервал нестрого может пониматься как один из фрагментов границы полиэдра, входящего во

множество решений ИСЛАУ (строгое определение см. в [29]). В общем случае этот фрагмент может представлять собой прямую, луч, отрезок, точку (см. рис. 1.2), а множество решений – политоп, граница которого составлена из указанных фрагментов.

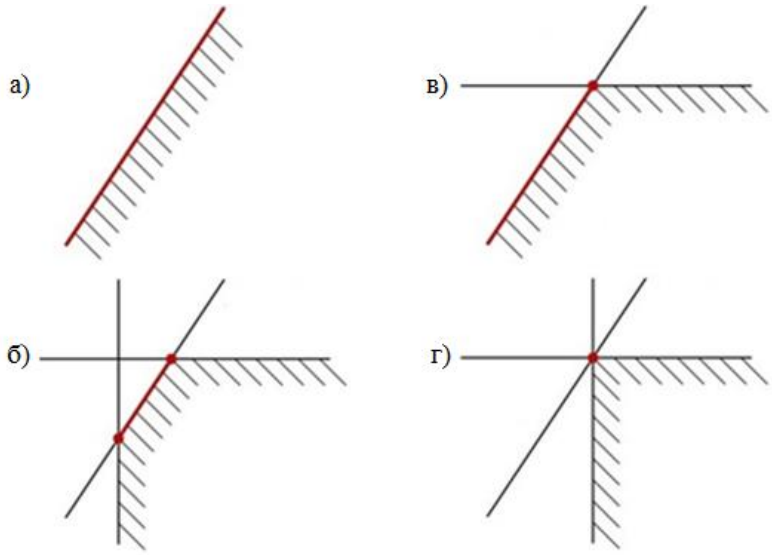


Рисунок 1.2. Виды граничных интервалов: а) прямая; б) отрезок; в) луч; г) точка

Для описания политопа метод граничных интервалов на первом шаге предписывает определённым образом вычислить координаты всех граничных интервалов по входным данным – матрице $\boldsymbol{A}$ и вектору $\boldsymbol{b}$ правой части ИСЛАУ. При этом из матрицы $\boldsymbol{A}$ и вектора $\boldsymbol{b}$ исключаются строки, которые в дальнейшем порождают граничные интервалы, оказывающиеся пустыми или совпадающими со всем пространством $\mathbb{R}^2$ или $\mathbb{R}^3$. На следующем шаге алгоритма необходимо построить замкнутый обход фрагментов границы полиэдров в каждой из координатных четвертей. Граница двумерного политопа представляет собой замкнутую ломаную с конечным числом звеньев. Замкнутость означает, что отправившись из произвольной вершины и двигаясь по звеньям ломаной, мы вернемся в исходную вершину. Отсутствие самопересечений говорит о том, что при таком движении политоп будет всегда оставаться по одну сторону. Одномерный политоп – это отрезок. Его замкнутым обходом будем считать движение по единственному ребру от произвольной вершины к противоположной и возвращение назад. Нульмерный политоп – это точка. Замкнутым обходом границы будем считать

последовательность из одного элемента – единственной вершины этого политопа. Такой обход не имеет звеньев (и ему не соответствует ломаная линия). Множество решений при этом может получиться одного из следующих видов:

- неограниченное;
- пустое;
- ограниченное;
- вырожденное до одного ребра;
- вырожденное до одной точки.

После получения упорядоченной последовательности фрагментов границ полиэдров с заданным координатным описанием, задача визуализации становится технической.

Выводы по главе 1

1. Интервальный анализ данных это востребованная область анализа данных, который представляет наивысшую ценность в задачах, где неопределённости и неоднозначности возникают с самого начала и являются неотъемлемой частью постановки задачи.

2. Необходима адаптация созданных функциональных узлов в системе `KNIME` для интервального анализа данных в рамках проекта `KNIME Interval Tools` под текущую версию системы `KNIME` и обновление библиотеки `JInterval`.

3. Есть потребность в разработке дополнительного функционального узла в системе `KNIME`, который пополнит проект `KNIME Interval Tools` узлом-визуализатором множеств решений интервальных систем линейных алгебраических уравнений в низкоразмерных случаях $\mathbb{R}^2$и $\mathbb{R}^3$.

ГЛАВА 2. ФУНКЦИОНАЛЬНЫЕ УЗЛЫ СИСТЕМЫ KNIME ДЛЯ ИНТЕРВАЛЬНОГО АНАЛИЗА ДАННЫХ

Настоящая глава посвящена разработке модуля, который реализует метод граничных интервалов для визуализации множеств решений интервальных систем линейных алгебраических уравнений в системе KNIME и адаптации ранее созданных функциональных узлов для интервального анализа данных описанных в главе 1.

2.1. Требования к разрабатываемому узлу

При разработке модуля, реализующего метод граничных интервалов для визуализации множеств решений интервальных систем линейных алгебраических уравнений, необходимо в первую очередь заботиться о том, что визуализация может использоваться вместе с другими модулями для обработки данных. Немало важно, чтобы этот узел мог продолжать поток сценария анализа данных в системе KNIME. К разрабатываемому функциональному узлу были предъявлены следующие требования:

- задание варианта визуализируемой системы ($\boldsymbol{Ax = b},\ \boldsymbol{Ax \leq b},\ \boldsymbol{Ax \geq b}$);
- задание требуемого варианта визуализируемого множества решений ИСЛАУ (объединенного, допускового, управляемого);
- на вход узлу подаются коэффициенты интервальной $m \times n$ −матрицы $\boldsymbol{A}$ и коэффициенты интервального вектор-столбца $\boldsymbol{b}$ размера m;
- на выходе узел должен строить интерактивное изображение множества решений ИСЛАУ, с возможностью вращения, масштабирования и т.п.

В реализуемом узле должны быть один входной и один выходной порт. Входной порт используется для получения интервальной системы линейных алгебраических уравнений, которую надо визуализировать. В визуализации, как правило, нет необходимости иметь выходной порт потому, что на выходе у узла

рисуется изображение, однако в данном узле такой порт включен, для того, чтобы он передавал данные построенных замкнутых обходов полиэдров дальше в сценарий KNIME.

Для технической визуализации была выбрана технология JavaFX [31] – платформа для создания RIA (Rich Internet Application), позволяет строить унифицированные приложения с насыщенным графическим интерфейсом пользователя для непосредственного запуска из-под операционных систем, работы в браузерах и на мобильных телефонах, в том числе, работающих с мультимедийным содержимым. Среди возможностей этой платформы можно отметить: кроссплатформенность, поддержка каскадных таблицей стилей, поддержка анимации компонентов, возможность работы и отображение 3D графики и очень многое другое.

2.2. Создание функциональных модулей среды KNIME

Модуль KNIME – это, прежде всего, модуль Eclipse, что позволяет использовать весь набор возможностей среды. Любой пользователь продукта может расширить функциональный набор, написав собственный модуль. Имеется подробное руководство по написанию модулей расширения (см. в [21]). Существующая платформа не должна “знать” о модулях расширения заранее, она просто предоставляет точки расширения, к которым присоединяются пакеты. Eclipse выступает как платформа для модулей, выполняет их и разрешает зависимости. Структура модуля с зависимостями описана в файле plugin.xml. Кроме того, сам модуль может обеспечивать точки расширения для других. Также каждый модуль может иметь класс, который наследуется от класса Eclipse Module. Этот класс выполняется один раз при запуске среды и вызывается автоматически.

Процесс создания нового узла KNIME подробно описан на официальном сайте продукта [21]. По этой причине в работе этот процесс не описывается подробно, а излагаются лишь некоторые необходимые этапы.

У каждого узла KNIME должна быть строго определённая сигнатура классов. Узел обязательно должен иметь следующие классы: NodeFactory, NodeDialogPane, NodeModel, NodeView рис 2.1. KNIME взаимодействует только с классом Node. Этот класс заботится о вложении узла в саму инфраструктуру системы KNIME. Новые узлы создаются с помощью классов: NodeModel, NodeView и NodeDialog.

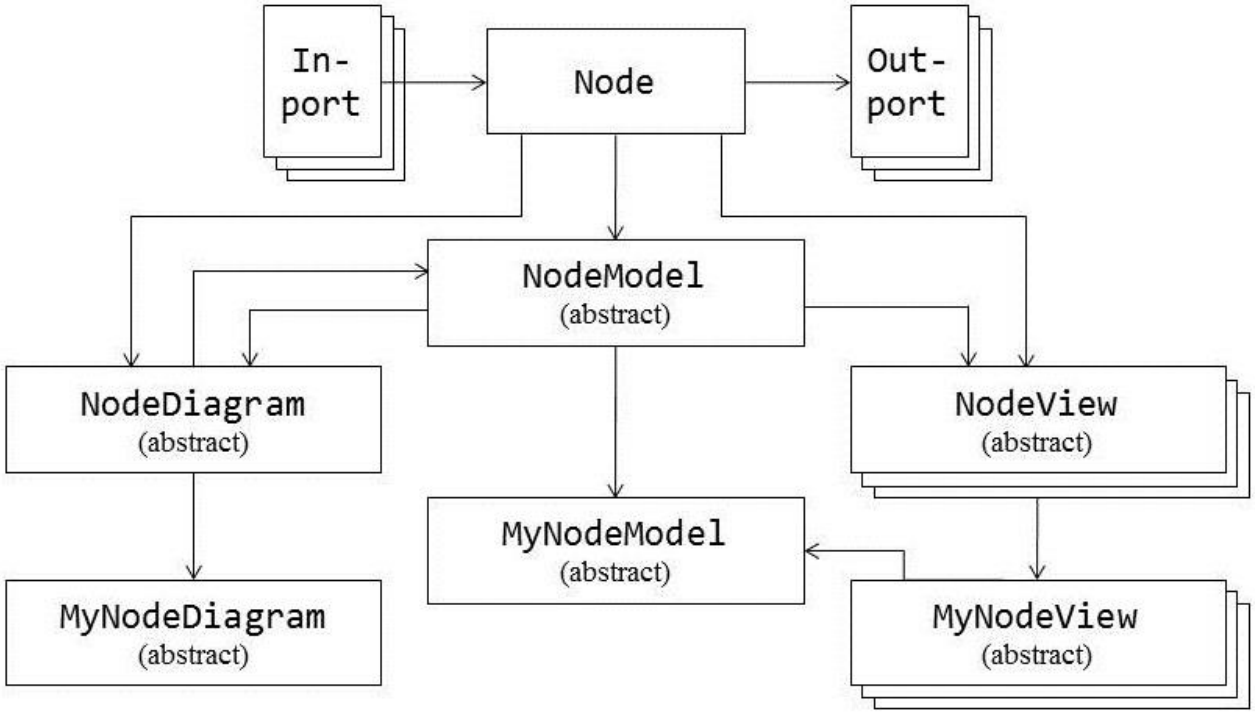


Рисунок 2.1. Схема взаимодействия классов в узле KNIME

NodeFactory – класс создающий объекты остальных классов. В нём обязательно должны быть реализованы методы:

- createNodeDialogPane() – метод возвращающий объект класса NodeDialogPane;
- createNodeModel() – метод возвращающий объект класса NodeModel;
- createNodeView(int, NodeModel) – возвращает представление данных в узле в зависимости от индекса;

- `getNrNodeViews()` – возвращает количество представлений уза, 0 если узел не имеет представлений;
- `hasDialog()` – возвращает `true` или `false` в зависимости от того можно ли настраивать узел.

Если узел требует специальной настройки пользователя, ему будет показано диалоговое окно настройки. Для этого необходимо реализовать `NodeDialogPane`, которая будут отображаться на платформе, когда пользователь дважды нажимает на узел или выберет пункт `configure` во всплывающем меню. Кроме того `NodeFactory` должен быть настроен для показа этой панели, то есть метод `hasDialog()` должен возвращать `true`. Некоторые диалоговые компоненты узел будет иметь по умолчанию.

`NodeModel` – это класс, в котором сосредоточен основной прикладной функционал узла. Конструктор класса в простейшем виде должен лишь вызывать родительский конструктор (см. листинг 2.1).

Листинг 2.1. Конструктор класса `NodeModel`

```
protected MyNodeModel() {
   super(1, 1);
}
```

Первый параметр родительского конструктора задаёт число входных портов узла, второй – количество выходных. Определённый набор методов класса заранее определён и должен быть реализован в соответствии с сигнатурой. Обязательной реализации подлежат не все из методов набора, а лишь некоторые. В частности, метод `configure(DataTableSpec[])` должен проверять входные данные на соответствие и настраивать узел под конкретные данные. Главная из всех функция – `execute(finalBufferedDataTable[], ExecutionContext)` – должна содержать описание выполняемых узлом действий и возвращает их результат (построенную модель, обработанные данные и т.п.).

Класс и его производные `NodeView` позволяет отображать данные `NodeModel` определенным образом. В `KNIME` есть довольно широкий набор

специальных классов для визуализации таблиц, графиков и т.п., который при желании может быть расширен различными специализированными технологиями, которые разработаны на `Java`.

2.3. Узел визуализации множеств решений ИСЛАУ

В соответствии с требованиями к разрабатываемому функциональному узлу, описанными в разделе 2.1. в состав узла-визуализатора включены методы, которые необходимы для выполнения основных шагов алгоритма граничных интервалов для визуализации интервальных систем линейных алгебраических уравнений [29]. Все представленные ниже методы используются только в классе `NodeModel` и имеют уровень доступа `private`. Рассмотрим подробное описание основных методов узла-визуализатора:

1. *Название*: `double[][] getBoundaryIntervals(double[][] A, double[] b)`.

   Описание: вычисляет матрицу граничных интервалов множества решений ИСЛАУ в ортантах.

   Входные данные: матрица A и вектор b.

   Выходные данные: матрица граничных интервалов, в которой первые 1-2 столбцы: (x, y) кординаты начала граничного интервала, 3-4 столбцы: (x, y) координаты конца граничного интервала и 5-й столбец: индекс граничного интервала.
2. *Название*: `double[][] getOrientationPoints(double[][] S)`.

   Описание: агрегирует координаты вершин полиэдров в матрицу.

   Входные данные: матрица граничных интервалов.

   Выходные данные: матрица координат вершин политопа: (x, y) в $\mathbb{R}^2$, (x, y, z) в $\mathbb{R}^3$.

3. *Название*: `double[][] getIntervals2Path(double[][] S)`.
   Описание: строит замкнутый обход граничных интервалов для множества решений в ортанте.
   Входные данные: матрица граничных интервалов.
   Выходные данные: матрица координат вершин полиэдра с построенным замкнутые обходом по часовой стрелки: (x, y) в $\mathbb{R}^2$, (x, y, z) в $\mathbb{R}^3$.
4. *Название*: `double[] ClearRowsMatrix(double[][] A, double[] b)`, `double[] ClearRowsVector(double[][]A, double[]b)`.
   Описание: удаляют из матрицы/вектора строки, заведомо дающие пустое множество решений, и строки, решение которых совпадает со всем пространством $\mathbb{R}^2$ или $\mathbb{R}^3$.
   Входные данные: матрица A и вектор b.
   Выходные данные: матрица/вектор без несовместных строк.
5. *Название*: `double[][] DrawingBox(double[][] V)`.
   Описание: определяет “размены” множества решений.
   Входные данные: матрица координат вершин политопа.
   Выходные данные: матрица, содержащая минимальные и максимальные координаты вершин множества решений ИСЛАУ.
6. *Название*: `double[] PathScene(double[][] P, double h)`.
   Описание: масштабирует замкнутые обходы полиэдров для лучшей наглядности так, чтобы они подходили под окно визуализации;
   Входные данные: матрица координат вершин полиэдра с построенным замкнутые обходом и коэффициент масштабирования h.
   Выходные данные: входная матрица с масштабированным замкнутым обходом полиэдра.
7. *Название*: `double[][] getOutPoints(double[][] V)`.
   Описание: формирует матрицу выходных данных содержащую построенные замкнутые обходы с заданным координатным представление вершин полиэдров в каждом ортанте. Данные

формируются в табличном формате, строки в таблице: $Q_i\_P_j\_V_k$, где $Q_i - i$-й ортант, $P_j - j$-й полиэдр, $V_k - k$-я вершина полиэдра, а столбцы: координаты (x, y) в $\mathbb{R}^2$, (x, y, z) в $\mathbb{R}^3$. рис 2.2. В пространстве $\mathbb{R}^2$ P_j отсутствует, так как полиэдр в ортанте всегда один.

Входные данные: матрица координат вершин полиэдра с построенным замкнутым обходом.

Выходные данные: матрица выходных данных.

Row ID	X	Y
Q1_V1	0	0
Q1_V2	2	2
Q1_V3	2	0
Q1_V4	0	0
Q2_V1	0	0
Q3_V1	0	-2
Q3_V2	0	0
Q3_V3	-0	-2
Q4_V1	0	-2
Q4_V2	0	0
Q4_V3	2	0
Q4_V4	4	-2
Q4_V5	0	-2

Row ID	X	Y	Z
Q1_P1_V1	0.286	0	0.286
Q1_P1_V2	0.286	0	0
Q1_P1_V3	0.286	0.286	0
Q1_P1_V4	0.286	0.286	0.286
Q1_P1_V5	0.286	0	0.286
Q1_P2_V1	0	0.286	0
Q1_P2_V2	0	0.286	0.286
Q1_P2_V3	0.286	0.286	0.286
Q1_P2_V4	0.286	0.286	0
Q1_P2_V5	0	0.286	0
Q1_P3_V1	0	0.286	0.286
Q1_P3_V2	0	0	0.286
Q1_P3_V3	0.286	0	0.286

Рисунок 2.2. Примеры окон с выходными данными: слева) в $\mathbb{R}^2$; справа) в $\mathbb{R}^3$

Выше представлены основные методы, но, их не достаточно для реализации алгоритма визуализации множеств решений ИСЛАУ. Поэтому, дополнительно, в составе узла используются и вспомогательные методы, например как:

1. *Название*: `double Max(double[][] V)`.

   Описание: находит максимальный элемент в матрице.

   Входные данные: двумерная матрица размерности $(m \times n)$.

   Выходные данные: значение максимального элемента во входной матрице.

2. *Название*: `double Sign(double x)`.

   Описание: определяет знак числа.

   Входные данные: число типа `double`.

Выходные данные: Если число положительное, функция возвращает значение 1. Если число отрицательное, функция возвращает значение -1. Если число равно 0, функция возвращает значение 0.

3. *Название*: `double MaxCol(double[][] V, int c)`, `double MinCol(double[][] V, int c)`.

   Описание: находит максимальный/минимальный элемент в матрице в заданном столбце.

   Входные данные: двумерная матрица и с-номер столбца.

   Выходные данные: максимальный/минимальный элемент во входной матрице в заданном столбце.

4. *Название*: `double CheckFinite(double[][] S)`.

   Описание: проверяет матрицу на наличие в строках символов бесконечности;

   Входные данные: двумерная матрица.

   Выходные данные: 0 – если матрица не имеет символов бесконечности, иначе количество строк с символами бесконечности.

5. *Название*: `double[][] DeleteFinite(double[][] V)`.

   Описание: удаляет строки в матрице, в которых присутствуют бесконечные элементы.

   Входные данные: двумерная матрица размерности $(m \times n)$.

   Выходные данные: входная матрица без строк, в которых присутствовали символы бесконечности.

6. *Название*: `double[] MinRowAbs(double[][] V, int k)`.

   Описание: находит минимальный по модулю элемент в матрице в определенной строке и его столбец.

   Входные данные: двумерная матрица и заданная строка k.

   Выходные данные: одномерный массив, где первый элемент: минимальный элемент во входной матрице, а второй элемент – номер столбца минимального элемента во входной матрице.

Список методов, которые необходимы для работоспособности функционального узла намного обширнее, выше представлены только основные методы, полный список модулей, входящих в узел-визуализатор, и их исходные коды см. в приложение.

2.4. Запуск визуализатора

При запуске узла вызывается метод `execute()`, который получает данные с входного порта в табличном виде. Сначала вызываются методы: `ClearRowsVector`, `ClearRowsMartix`, которые для каждого ортанта будущего политопа удаляют из системы строки, которые заведомо имеют пустое множество решений, и строки, решение которых есть все пространство $\mathbb{R}^2$ или $\mathbb{R}^3$. После этого метод `getBoundaryIntervals` вычисляет матрицу граничных интервалов множества решений ИСЛАУ. Если в этой матрице нет ни одной строки, то множество решений пусто (нет граничных интервалов). Затем все координаты вершин полиэдров собираются в матрицу методом `getOrientationPoints`, из которой, в этом же методе удаляются одинаковые строки методом `NonRepeatRows`. На следующем шаге вызывается метод `DrawingBox`, который определяет брус, в котором будет отображаться множество решений ИСЛАУ. Затем методом `getIntervals2Path` происходит построение замкнутых обходов (по часовой стрелки) множества решений для каждого ортанта. После этого метод `PathScene` масштабирует замкнутые обходы под окно визуализации для лучшей наглядности. Далее методом `getOutPoints` данные собираются в нужном выходном формате. Заключительным этапом происходит отображения множества решений технологией `JavaFX`. В листинге 2.2. показан пример добавления 4-х многоугольников P_i, где i – номер ортанта, в окно визуализации `JavaFX` в пространстве $\mathbb{R}^2$, полный код см. в приложении.

Листинг 2.2. Добавление многоугольников в окно визуализации JavaFX в узле-визуализаторе в пространстве $\mathbb{R}^2$

```
public class ILSVisualizerNodeView
    extends NodeView<ILSVisualizerNodeModel> {

    private Polygon createStartingPolygon(double[] a) {
        Polygon poly = new Polygon(a);
        return poly;
    }
    …
    protected void modelChanged() {
        final Group Group = new Group();

        double [] P1 = nodeModel.getP1();
        double [] P2 = nodeModel.getP2();
        double [] P3 = nodeModel.getP3();
        double [] P4 = nodeModel.getP4();

        final Polygon poly1 = createStartingPolygon(P1);
        final Polygon poly2 = createStartingPolygon(P2);
        final Polygon poly3 = createStartingPolygon(P3);
        final Polygon poly4 = createStartingPolygon(P4);

        Group.getChildren().addAll(poly1, poly2, poly3, poly4);
        …
    }
    …
}
```

Методом `get()` из класса `NodeModel` данные построенных замкнутых обходов поступают в класс `NodeView`, и в дальнейшем визуализируются. Для создания многоугольников используется метод `createStartingPolygon` – создает объект типа `Polygon`. В узле визуализации используются стандартные классы `JavaFX`, полное описание каждого класса см. в [32]:

1. *Название*: `Polygon`.

   Описание: создает многоугольник, определяемый массивом, в котором содержатся (x, y) координаты вершим замкнутого обхода.

   Конструктор: `Polygon()` – создает пустой экземпляр многоугольника;

2. *Название*: `Line`.

   Описание: создает линию, представляет собой отрезок из (x, y) координат начала и конца.

Конструктор: `Line(double startX, double startY, double endX, double endY)` – создает новый экземпляр линии;

3. *Название*: `Text`.

   Описание: определяет узел, который выводит на экран текст.

   Конструктор: `Text(double x, double y, String text)` – создает экземпляр текста на данных координатах, содержащих строку;

4. *Название*: `MeshView`.

   Описание: определяет поверхность с заданными данными 3D сетки.

   Конструктор: `MeshView(Mesh mesh)` – создает экземпляр класса `MeshView` с указанной поверхность `Mesh`;

5. *Название*: `TriangleMesh`.

   Описание: расширяет абстрактный класс `MeshView` и определяет 3D геометрический объект, который представляет триангулированную геометрическую сетку.

   Конструктор: `TriangleMesh()` – создает экземпляр класса `TriangleMesh`;

6. *Название*: `Box`.

   Описание: определяет 3-мерный куб заданного размера. Куб представляет собой 3D-геометрический примитив, создается с заданной длиной, шириной и высотой.

   Конструктор: `Box(double width, double height, double depth)` – создает экземпляр куба с заданной длиной, шириной и высотой.

2.5. Адаптация созданных ранее узлов в KNIME

Библиотека `JInterval` и функциональные узлы для интервального анализа данных `KNIME Interval Tool` находятся в репозиториях [20, 33]. Для того чтобы можно было редактировать узлы и использовать библиотеки была произведена их выгрузка, сборка и компиляция с помощью дополнительное

расширение для Eclipse, которое называется Subversion [34]. После успешного выполнения этих операций дополнительно скачивался и подключался в папку библиотек проекта jar-файл lpsolve. Для запуска KNIME из под Eclipse было необходимо в файле MANIFEST, который имеет расширение MF подключить все используемые узлами библиотеки. При попытке компиляции среда сразу указала на наличие ошибок в коде, произошло это из-за того, что используемые в узле библиотеки уже используют другие конструкторы, т.е. на вход они принимают данные в других форматах, нежели раньше. Класс NodeModel в узле ILS Solver больше всего нуждался в исправлениях. Класс net.java.jinterval.ils.IntervalMatrix, который работает с интервальными матрицами и векторами, а так же класс net.java.jinterval.ils в нынешнем виде стали более просты и удобны. Изменились некоторые функции, например: numsToInterval() – преобразует число в интервал. Так же поменялись многие конструкторы, например: GaussSeidelIterativeSolver() – отыскивает методом Гаусса-Зейделя оценку объединенного множества решений интервальных систем линейных алгебраических уравнений, SubdiffNewtonIterativeSolver() – находит формальные решения ИСЛАУ в арифметике Каухера и т.п. Пример изменений класса NodeModel см. листинг 2.3.

Листинг 2.3. Фрагмент изменений узла ILS Solver

```
public class ILSSolverNodeModel extends NodeModel {
    …
    protected BufferedDataTable[] execute(final BufferedDataTable[]
              inData, final ExecutionContext exec) throws Exception {
        …
        SetIntervalContext ctx
        SetIntervalContexts.getInfSup(BinaryValueSet.BINARY64);
        IntervalMatrix A = new IntervalMatrix(n, n);
        IntervalVector b = new IntervalVector(n);
        IntervalVector x = new IntervalVector(n);

        if(m_solution_type.getStringValue().equals(ST_OUTER_UNITED)){
            IntervalVector x0 = new IntervalVector(n);
            x0.set( (SetInterval) ctx.numsToInterval(-10000, 10000) );
            GaussSeidelIterativeSolver solver = new
            GaussSeidelIterativeSolver(BinaryValueSet.BINARY64, 16);
```

```
            x = solver.solve(A, b, x0);
            for(int i=0; i<n; i++){
                xinf[i] = x.getEntry(i).doubleInf();
                xsup[i] = x.getEntry(i).doubleSup();
            }
        }
        if(m_solution_type.getStringValue().equals(ST_INNER_UNITED)){
            SubdiffNewtonIterativeSolver solver = new
            SubdiffNewtonIterativeSolver(BinaryValueSet.BINARY64, 10);
            x = solver.solve(A, b);
            for(int i=0; i<n; i++){
                xinf[i] = x.getEntry(i).doubleInf();
                xsup[i] = x.getEntry(i).doubleSup();
            }
        }
        if(m_solution_type.getStringValue().equals(ST_INNER_TOLERABLE)){
            double[][] Amid = new double[n][n];
            for (int i = 0; i < Ainf.length; i++) {
            for (int j = 0; j < Ainf[0].length; j++) {
                Amid[i][j]=(Ainf[i][j]+Asup[i][j])/2;
                }
            }
            double[] bmid = new double[n];
            for (int j = 0; j < binf.length; j++) {
                bmid[j]=(binf[j]+bsup[j])/2;
            }
            RealMatrix coef = new Array2DRowRealMatrix(Amid);
            DecompositionSolver calc = new
            LUDecomposition(coef).getSolver();
            RealVector right = new ArrayRealVector(bmid);
            RealVector h = calc.solve(right);
            double[] ans = h.toArray();
            ExtendedRational [] y = new ExtendedRational[n];
            for (int j = 0; j < n; j++) {
                y[j] = ExtendedRational.valueOf(ans[j]);
            }
            ShaidurovTolBoxDimention solver = new
            ShaidurovTolBoxDimention(A, b, y);
            x = solver.getBrus();
            for(int i=0; i<n; i++){
                xinf[i] = x.getEntry(i).doubleInf();
                xsup[i] = x.getEntry(i).doubleSup();
            }
        }
        …
        container.close();
        BufferedDataTable out = container.getTable();
        return new BufferedDataTable[]{out};
    }
}
```

Аналогичные действия, связанные с изменениями передаваемых данных в конструкторы и некими изменениями методов классов были проделаны и с другими функциональными узлами. Полный исходный код изменений см. приложение.

Выводы по главе 2

1. Произведена адаптация разработанных функциональных узлов для интервального анализа данных в рамках проекта `KNIME Interval Tool` под текущие обновления платформы `KNIME` и библиотеки `JInterval`.

2. Разработан и включен в пакет `KNIME Interval Tool` узел `ILS Visualizer` для визуализации множеств решений интервальных систем линейных алгебраических уравнений в низкоразмерных случаях ($\mathbb{R}^2$ и $\mathbb{R}^3$).

3. Узлы в своем новом варианте позволят создавать сценарии анализа с возможностью построения интервальной регрессии и визуализации ИСЛАУ в комплексе с уже имеющимися стандартными методами системы `KNIME`.

ГЛАВА 3. РЕШЕНИЕ ПРИКЛАДНЫХ ЗАДАЧ ИНТЕРВАЛЬНОГО АНАЛИЗА ДАННЫХ В СИСТЕМЕ KNIME

В настоящей главе описаны примеры работы созданного узла-визуализатора и адаптированных функциональных узлов для проверки работоспособности разработанных программных инструментов.

3.1. Пример работы пользователя с узлом-визуализатором

Для примера, проведем работу пользователя, при использовании узла для визуализации интервальной системы линейных алгебраических уравнений в простом сценарии системы KNIME рис. 3.1.

Система: $\begin{pmatrix} 1 & [0,1] \\ [0,1] & [-4,-1] \end{pmatrix} x = \begin{pmatrix} [0,2] \\ [0,2] \end{pmatrix}$

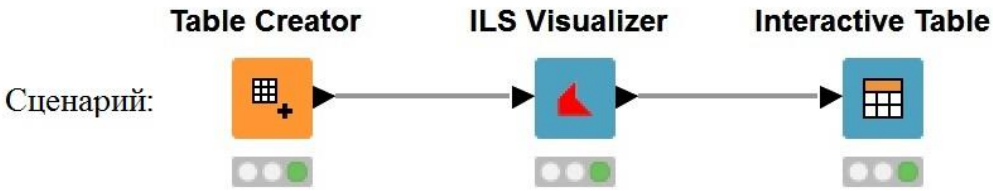


Рисунок 3.1. Сценарий KNIME и использованием узла визуализатора

Перед запуском визуализатора в пакете KNIME, ему нужно подать на вход узел, в котором будет в определенном формате записана ИСЛАУ рис 3.2.

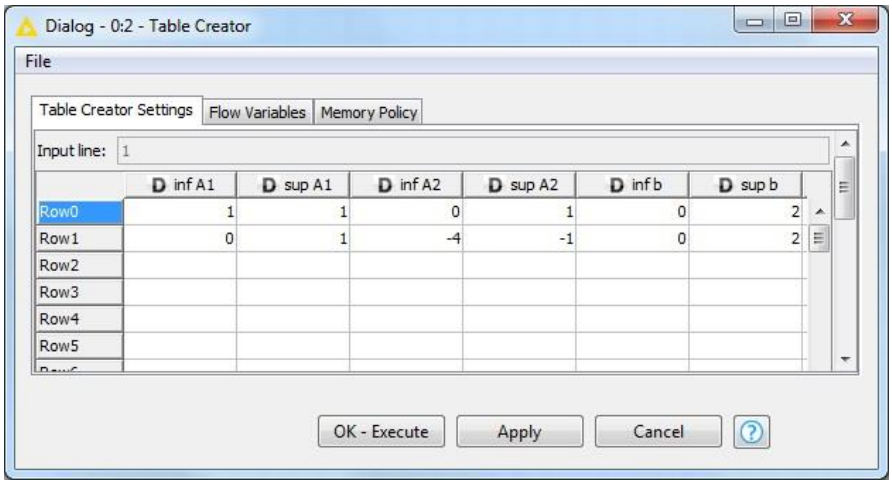


Рисунок 3.2. Таблица с входными данными

После этого необходимо задать настройки узла, а именно: выбрать тип входной системы и тип множества решений. Предположим, что мы решили визуализировать объединенное множества решений системы $\boldsymbol{Ax = b}$ рис. 3.3.

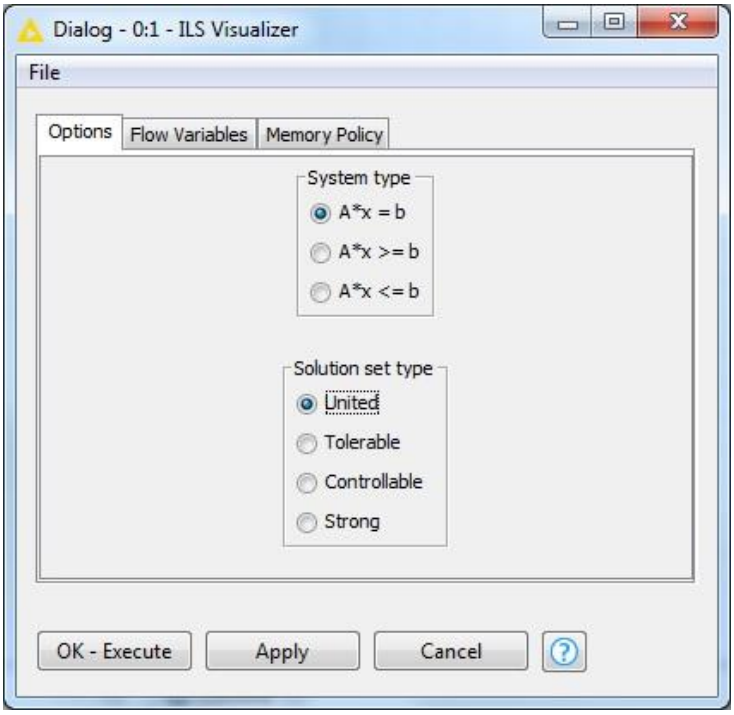


Рисунок 3.3. Конфигурирования узла

После запуска визуализатора становятся доступна опция: "`View: Sets of Solutions`" – отображение множества решений рис. 3.4. Для удобства пользователя имеется некий набор инструментов, который позволяет включить/выключить отображение координатной оси и/или полиэдров в любом из ортантов, например, в рис. 3.5 отключен 4-й ортант. Также доступен раздел "`Help`", в котором описаны возможности предоставляемые окном визуализации, например, есть возможность масштабирования, трехмерные множества решений можно вращать, множества решений, граничные интервалы которых уходят в бесконечность имеют границу с концом оси координат. Для того чтобы полностью быть уверенным в том, что множество имеет ребро уходящее в бесконечность, всегда можно посмотреть координаты в таблице выходных данных (см. ниже).

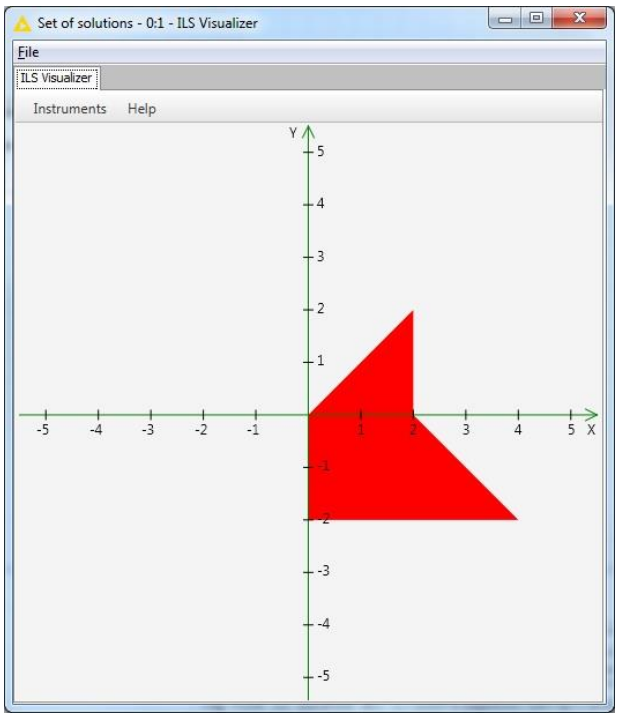


Рисунок 3.4. Визуализация множества решений ИСЛАУ

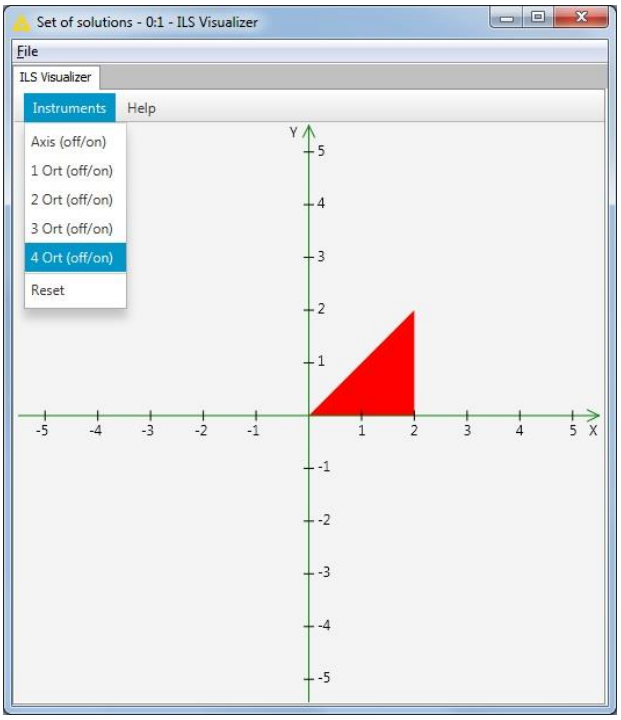
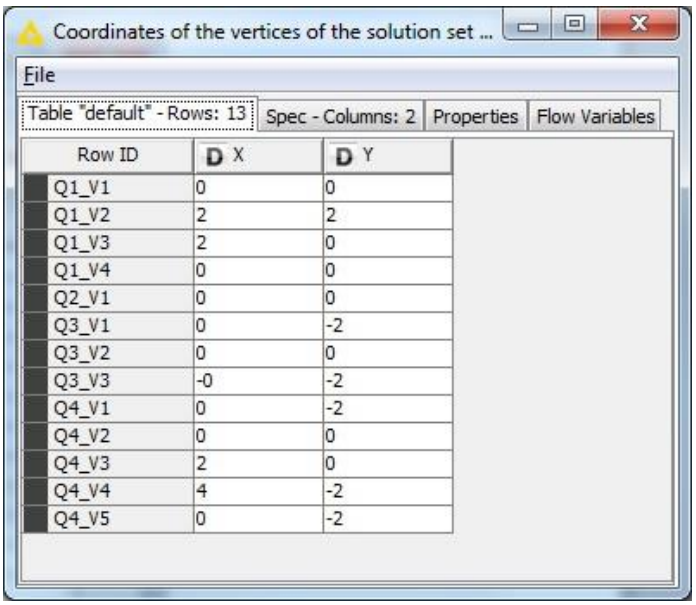


Рисунок 3.5. Пример работы инструмента

Помимо визуализации узел опцией: “`Coordinates of the vertices of the solution set`” – позволяет увидеть полученный замкнутый обход с заданным координатным представлением полиэдров в каждом ортанте рис 3.6.



Row ID	X	Y
Q1_V1	0	0
Q1_V2	2	2
Q1_V3	2	0
Q1_V4	0	0
Q2_V1	0	0
Q3_V1	0	-2
Q3_V2	0	0
Q3_V3	-0	-2
Q4_V1	0	-2
Q4_V2	0	0
Q4_V3	2	0
Q4_V4	4	-2
Q4_V5	0	-2

Рисунок 3.6. Таблица выходных данных узла визуализатора

Полученные таким образом замкнутые обходы можно дальше передавать в сценарии системы KNIME для различного рода исследований.

3.2. Примеры работы узла визуализатора с ИСЛАУ

Для демонстрации работы узла визуализатора, множества решений интервальных систем линейных алгебраических уравнений в данном разделе взяты из известных источников [1-2, 35-37]. Примеры см. рисунки: 3.7-3.9.

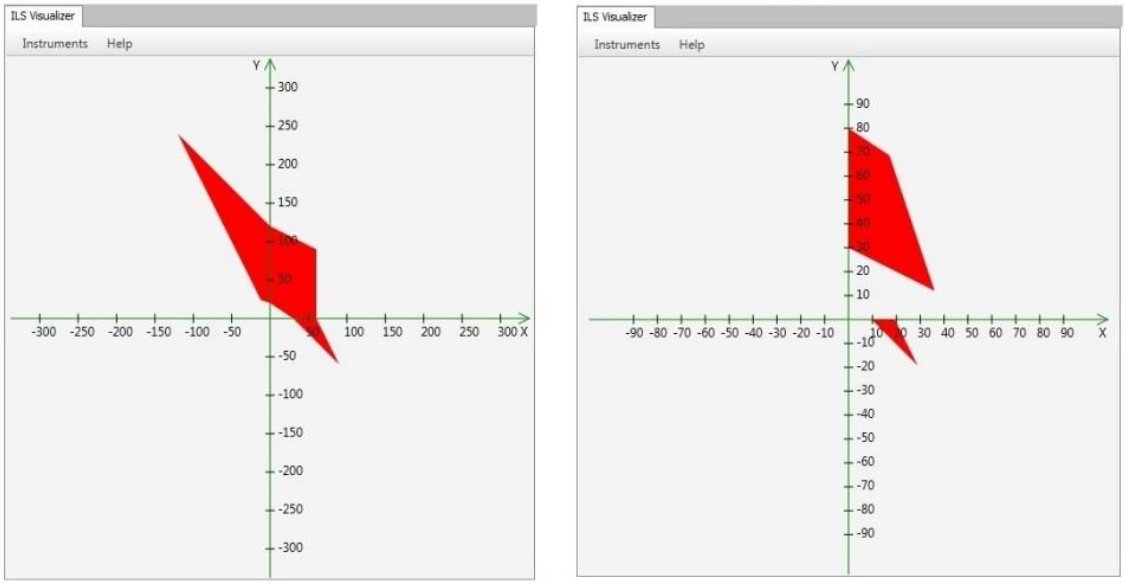


Рисунок 3.7. Визуализация множеств решений системы Хансена:

слева) объединенное; справа) допусковое

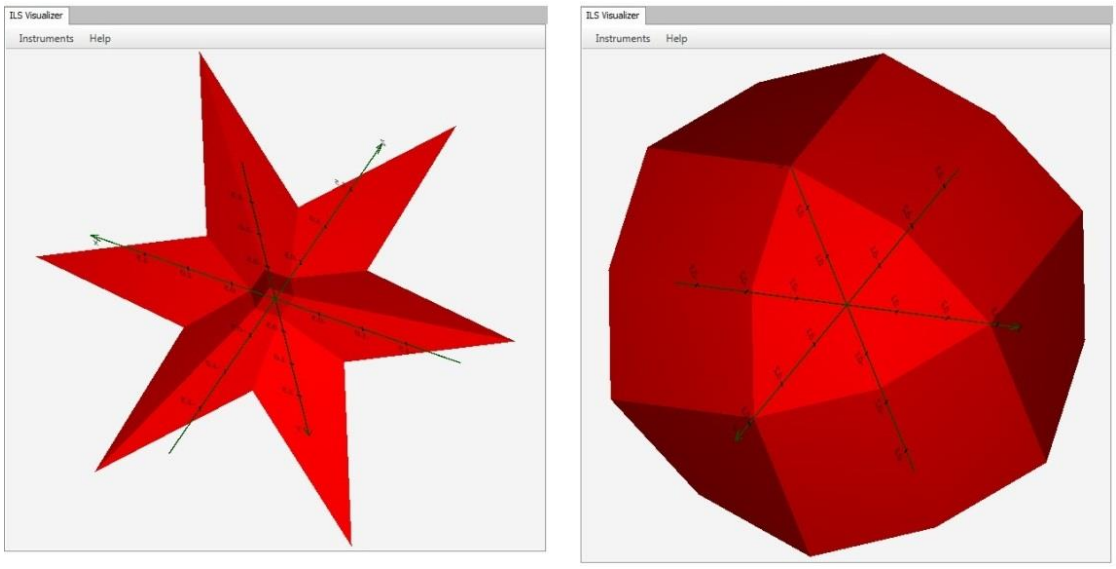


Рисунок 3.8. Визуализация множеств решений системы Ноймайера: слева) объединенное; справа) допусковое

Рисунок 3.9. Визуализация объединенного множества решений системы Ноймайера, в частности с применением инструментов узла

На рисунке 3.10 показано допусковое и управляемое множество решений ИСЛАУ $([-1,1]\ [1,-1])\ x = ([-1,1])$. Границы управляемого множества решений уходят в бесконечность.

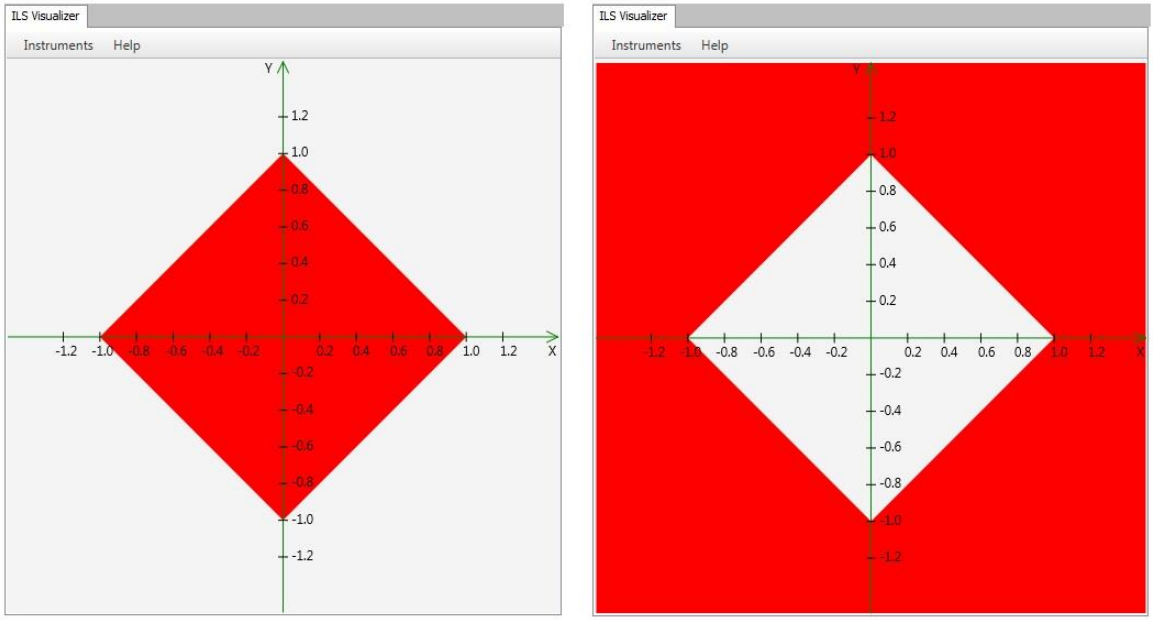


Рисунок 3.10. Визуализация множеств решений системы $([-1,1]\ [1,-1])\ x = ([-1,1])$: слева) допусковое; справа) управляемое

Корректность работы узла проверена многочисленными тестами, в том числе, приводящим к вырожденным, пустым и прочим особым вариантам множеств решений ИСЛАУ.

3.3. Решение прикладных задач с помощью функциональных узлов

Для демонстрации работы функциональных узлов выбрана задача электрической цепи см. главу 1. На примере электрической цепи, заимствованном из [38], показано, каким образом задача о допусках может быть решена с использованием несложного сценария системы `KNIME` с использованием пакета `KNIME Interval Tools`.

Информация о структуре цепи отражается в уравнениях, выражающих первый и второй законы Кирхгофа, которые определяют баланс токов в сечениях цепи и баланс напряжений в контурах соответственно. Для линейной электрической цепи, состоящей из несвязанных резисторов и независимых

источников питания с m ветвями и $(n+1)$ узлами (один из которых – заземление), первый закон Кирхгофа дает уравнения

$$\sum_{j=1}^{m} m_{kj} i_j = 0,\ k = 1, \dots, n,$$

где i_j – сила тока в $j-$й ветви цепи, а α_{kj} – элемент матрицы инцидентности $\alpha = (a_{kj})_{m \times n}$ задающей структуру цепи. Значения элементов матрицы инцидентности определяются по следующему правилу:

$$a_{kj} = \begin{cases} -1, & \text{если ток течет в } k-\text{й узел по } j-\text{й ветви}, \\ 0, & \text{если } j-\text{я ветвь не связана с } k-\text{м узлом}, \\ 1, & \text{если ток течет от } k-\text{го узла по } j-\text{й ветви}. \end{cases}$$

Баланс напряжений в контурах цепи, определяемый вторым законом Кирхгофа, описывается уравнениями

$$r_p i_p - (V_{s_p} - V_{p_p}) = u_p, p = 1, \dots, m,$$

где r_p – сопротивление, i_p – сила тока, s_p и p_p – концевые узлы $p-$й ветви цепи, V_{s_p} и V_{p_p} – напряжение в узлах s_p и p_p, а u_p – напряжение в ветви, обеспечиваемое независимым источником. Первый и второй законы Кирхгофа, в совокупности могут быть переписаны как система $n+m$ уравнений с $n+m$ неизвестными $Ax = b$, где

$$A = \begin{bmatrix} R & a^T \\ a & 0 \end{bmatrix}, \qquad x = \begin{bmatrix} i \\ u \end{bmatrix}, \qquad b = \begin{bmatrix} u \\ 0 \end{bmatrix}, \qquad R = \begin{bmatrix} r_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & r_m \end{bmatrix}.$$

Схема цепи показана на рисунке 3.11. Резисторы $r_1, \dots, r_{11}$ на ветвях цепи для простоты полагаются одинаковыми с интервальными допусками на их сопротивления, равными [98, 102] Ом. Напряжения на источниках тока e_1 и e_2 варьируются в интервале [99, 101] В, а на источниках e_5 и e_7 – равны 10 В. Требуется отыскать границы диапазонов значений силы тока в каждой из одиннадцати ветвей цепи и напряжения в пяти ее узлах.

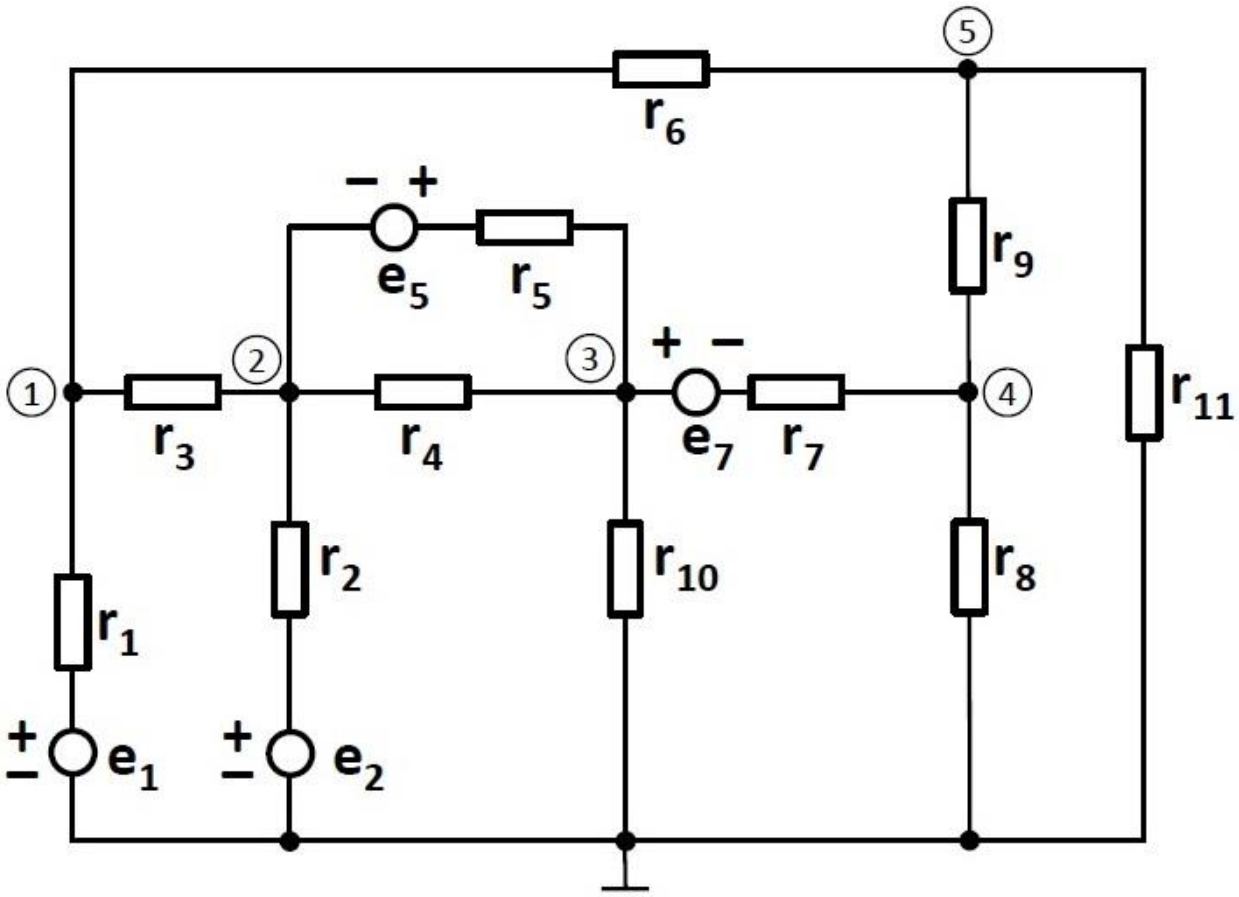


Рисунок 3.11. Схема электрической цепи

Построение и решение данной задачи с использованием разработанных модулей расширения платформы KNIME осуществляется с помощью простого сценария обработки данных, приведённого на рисунке 3.12.

Рисунок 3.12. Сценарий обработки данных в системе KNIME

Фрагмент входной таблицы, которая формируется коэффициентами интервальной матрицы ***A*** и вектором вектор-столбцом ***b*** см. рис 3.13.

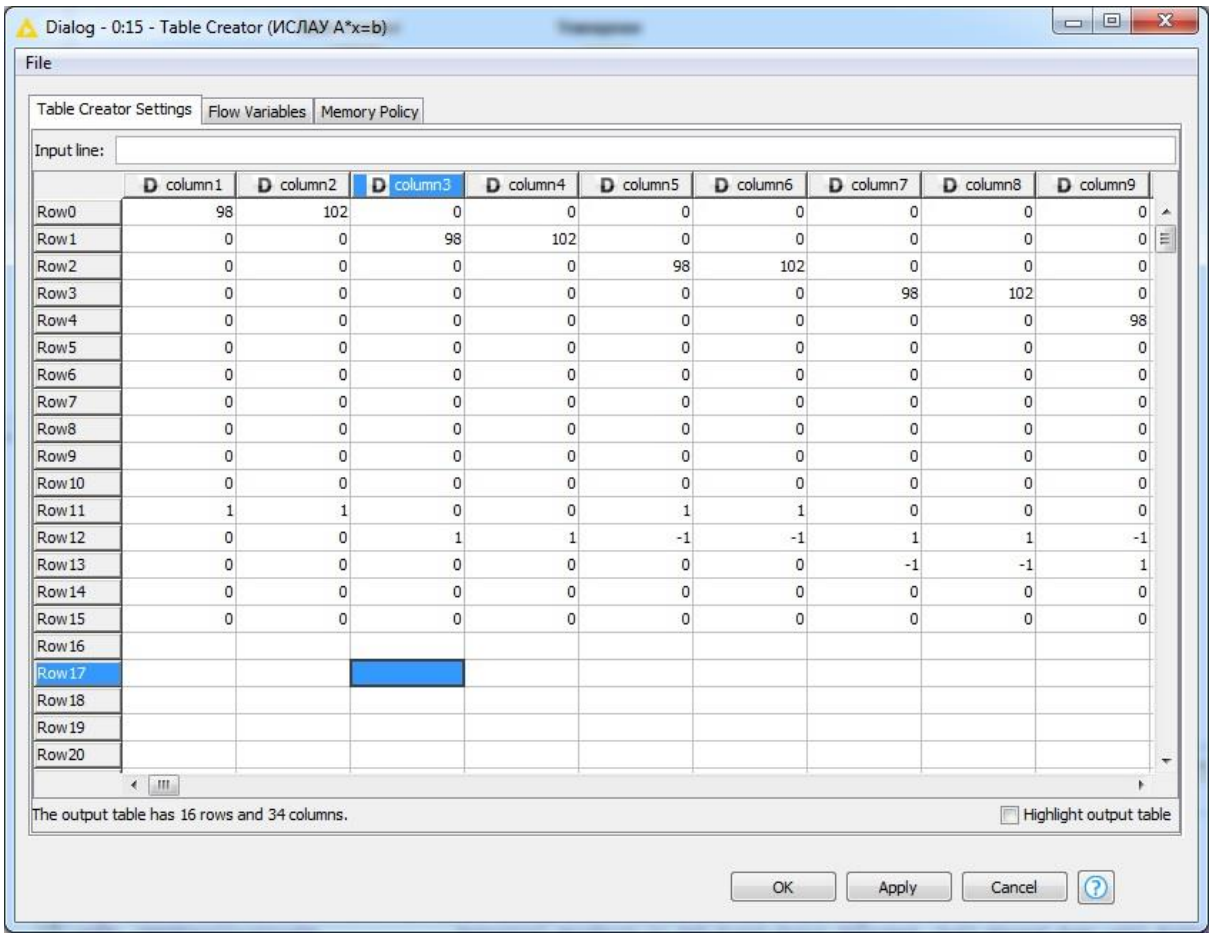


Рисунок 3.13. Фрагмент таблицы с входными данными дл узла `ILS Solver`

На рисунке 3.14 показаны настройки узла `ILS Solver` для решения поставленной задачи и результаты его работы.

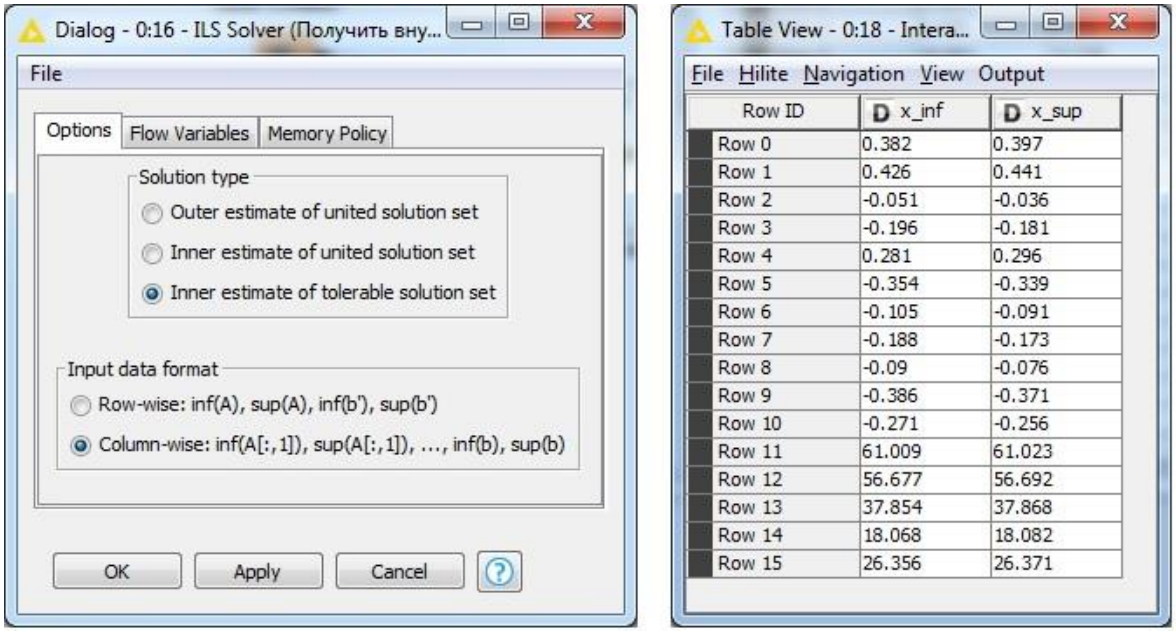


Рисунок 3.14. Работа с узлом `ILS Solver`: слева) конфигурация узла; справа) выходные данные

Сценарий в системе KNIME выдал значение интервалов допусков на силу тока в ветвях цепи см. таблица 3.1, и значение интервалов допусков на напряжение в узлах цепи см. таблица 3.2.

Таблица 3.1

Интервалы допусков на силу тока в ветвях цепи

Номер ветви цепи	X_inf	X_sup
1	0.3824879289756502	0.3971912154093764
2	0.4258034369970406	0.4405067234307668
3	-0.050667151238253515	-0.03596386480452731
4	-0.19558693733451013	-0.18088365090078393
5	0.28088365090078393	0.29558693733451014
6	-0.3538757073879861	-0.3391724209542599
7	-0.10521260578370804	-0.09050931934998184
8	-0.18810030631846741	-0.1733970198847412
9	-0.09023934375162246	-0.07553605731789625
10	-0.38596126888531235	-0.37125798245158614
11	-0.27098800685322677	-0.25628472041950057

Таблица 3.2

Интервалы допусков на напряжение в узлах цепи

Номер узла цепи	X_inf	X_sup
1	61.008691137531805	61.023394423965534
2	56.677140335392764	56.69184362182649
3	37.85361092362806	37.86831421006179
4	18.067514666943566	18.082217953377295
5	26.356284720419502	26.37098800685323

Выходные данные работы узла очень совпадают с основными результатами, полученными в [38]. Разработанные функциональные узлы протестированы и на других прикладных задачах см. главу 1. Результаты, полученные при решении интервальных задач с использованием пакета разработанных узлов, свидетельствуют об их опытной эксплуатации.

Выводы по главе 3

1. Показан пример работы пользователя с узлом-визуализатором (ILS Visualizer). Продемонстрированы примеры визуализации множеств решений известных интервальных систем линейных алгебраических уравнений. Результаты изображений множеств решений ИСЛАУ совпадают с результатами, выдаваемыми продуктом в MatLab [30] и источниками литературы [1, 2, 35].

2. С помощью созданного пакета функциональных узлов системы KNIME, для тестирования корректности адаптации узлов, решены прикладные задачи интервального анализа данных.

3. Разработанные в ходе настоящей работы функциональные узлы могут быть использованы в прикладных и учебных задачах анализа данных.

ЗАКЛЮЧЕНИЕ

В ходе настоящей работы был разработан пакет функциональных узлов для интервального анализа данных в системе KNIME.

Результаты работы могут быть сформулированы следующим образом:

1. Проанализированы существующие узлы для интервального анализа данных в рамках проекта KNIME Interval Tools в системе KNIME;

2. Произведена адаптация модулей расширения платформы KNIME, позволяющих исследовать данные с помощью метода интервальной регрессии, отыскивать выбросы в данных (IR Outlier Detector), строить модели (IR Learner), предсказывать отклик регрессии (IR Predictor), решать задачи линейного программирования (ILS Solver), проверять является ли данные для интервала регрессии совместимыми (IR Consistency);

3. Разработан узел (ILS Visualizer) для визуализации множеств решений ИСЛАУ, который пополнил проект KNIME Interval Tools;

4. Проведена опытная эксплуатация разработанных и адаптированных программных инструментов в сценариях KNIME, свидетельствующая о работоспособности созданного пакета.

Созданный пакет функциональных узлов для интервального анализа данных на аналитической платформе KNIME позволит применять метод интервального анализа данных не только самостоятельно, но и в совокупности с уже имеющимся набором не интервальных инструментов.

Основные положения и отдельные результаты работы докладывались и обсуждались на Всероссийской конференции “Математика и её приложения: фундаментальные проблемы науки и техники” (Барнаул, 2015), третьей Региональной конференции “Мой выбор – НАУКА!” (Барнаул, 2016), и опубликованы в [39].

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Шарый С. П. Конечномерный интервальный анализ. – Новосибирск: XYZ, 2014.
2. Шарый С. П. Разрешимость интервальных линейных уравнений и анализ данных с неопределенностями, Автомат и телемех., 2012, выпуск 2, 111–125.
3. Канторович, Л. В. О некоторых новых подходах к вычислительным методам и обработке наблюдений. Сиб. мат. журн., 3, 701 (1962).
4. Жилин С. И. Нестатистические модели и методы построения и анализа зависимостей: дис. канд. физико-математ. наук. Алтайский. гос. университет, Барнаул, 2004
5. Максимов А. В., Оскорбин Н.М. Многопользовательские информационные системы: основы теории и методы исследования: монография. – Барнаул: Изд-во Алтайского университета, 2005. — 250с.
6. Акулич И. Л. Задачи линейного программирования// Математическое программирование в примерах и задачах. – М.: Высшая школа, 1986. – 319 с.
7. Оскорбин Н. М. Некоторые задачи обработки информации в управляемых системах // Синтез и проектирование многоуровневых иерархических систем. Материалы конференции. Барнаул: Алтайский государственный университет, 1983.
8. Beard K., Buttenfield B., Clapham S. Visualization of Spatial Data Quality. Technical Paper 91-26, 1991.: Tech. Rep.: Castine, Maine: National Center for Geographic Information and Analysis, 1991.
9. Chil J.-P., Delfiner P. Geostatistics: Modeling Spatial Uncertainty. Wiley-Interscience, 1999. 720 p.
10. Ehlschlaeger C., Goodchild M. Uncertainty in Spatial Data: Defining, Visualizing, and Managing Data Errors // Proc. of Conference GIS/LIS. Phoenix: 1994. P. 246–253.

11. Автоматизация построения экологических карт. / Т. В. Байкалова, А. В. Евтюшкин, С. И. Жилин [и др.] // Методы дистанционного зондирования и ГИС-технологии для контроля и диагностики состояния окружающей среды: Тезисы докладов 3-й Международной конференции. Москва: МИИГАиК, 1996.
12. Информационные технологии автоматизации построения экологических карт. / Т. В. Байкалова, А. В. Евтюшкин, С. И. Жилин [и др.] // Проблемы предотвращения деградации земель Западной Сибири и осуществление государственного контроля за их использованием и охраной: Сб. науч. тр. Барнаул: Минсельхозпрод РФ, 1997. С. 108–113.
13. Prolubnikov A. An Interval Approach to Pattern Recognition of Numerical Matrices // Reliable Computing. 2013. Vol. 19, no. 1. P. 107–119.
14. Пролубников А. В. Интервальный подход к решению задачи распознавания числовых матриц // Вычислительные технологии. 2012. Т. 17, № 4. С. 77–87.
15. Kolev L. Interval Methods for Circuit Analysis. World Scientific Publishing Co. Pte. Ltd, 1993. 322 p.
16. Шайдуров В. В., Шарый С. П. Решение интервальной алгебраической задачи о допусках. (Препринт / ВЦ СО АН СССР; №5). Красноярск, 1988. 27 с.
17. Добронец Б. С., Шайдуров В. В. Двусторонние численные методы. Новосибирск: Наука, 1990.
18. Interval and Related Software [Электронный ресурс] / Ред. Kreinovich V. – Электрон. дан. – El Paso, 2009. – Режим доступа: http://www.cs.utep.edu/interval-comp/intsoft.html, свободный. – Загл. с экрана.
19. Pryce J., Keil C. (Tech Eds.) P1788: IEEE Draft Standard for Interval Arithmetic, Version 8.1.
20. JInterval. Java library for interval computations. [Электронный ресурс]. – Заголовок с экрана. Режим доступа: www.jinterval.java.net.

21. KNIME. [Электронный ресурс]. – Заголовок с экрана. Режим доступа: www.knime.org.
22. RapidMiner. [Электронный ресурс]. – Заголовок с экрана. Режим доступа: www.rapidminer.com.
23. Жилин С. И. Расширения системы KNIME для построения и анализа регрессионных зависимостей при интервальной ошибке // Материалы тринадцатой конф. по матем. “МАК-2010”, Барнаул, 2010. С. 3–4.
24. Zhilin S. KNIME extension for building multivariate regression under interval error // Seventh Winter Symposium on Chemometrics, 15–19 February 2010, St. Petersburg, Russia, – P. 63–64.
25. Данилов М. В., Дронов К. С., Жилин С. И. Решатель интервальных систем линейных уравнений для платформы KNIME // Труды Междунар. конф. “Современные проблемы прикладной математики и механики: теория, эксперимент и практика”, посв. 90-летию со дня рожд. ак. Н.Н. Яненко (Новосибирск, Россия, 30 мая – 4 июня 2011 г.), No. гос. регистр. 0321101160, ФГУП НТЦ "Информрегистр". – Новосибирск. – 2011. – URL:conf.nsc.ru/files/conferences/niknik-90/fulltext/38151/46754/Dronov.pdf.
26. Жилин С. И., Ледомский П.А. Модуль библиотеки JInterval для анализа интервальных данных методом главных компонент // Modelare matimatica, optimizare si tehnologii informationale: Materialele Conf. Intern.: Ed. a 4-a, 25 –28 mar. 2014, Chisinau / red. resp.: Dumitru Solomon. – Chisinau: Evrica, 2014 (Tipografia A ¸SM), – Vol. 2. – 2014. – P. 187–196.
27. Хемди А. Таха Глава 3. Симплекс-метод // Введение в исследование операций = Operations Research: An Introduction. – 7-е изд. – М.: Вильямс, 2007.
28. Kramer W. Computing and visualizing solution sets of interval linear systems // Serdica J. Computing. – 2007. – V. 1. – P. 455–468.
29. Шарая И. А. Метод граничных интервалов для визуализации полиэдральных множеств решений // Вычислительные технологии. – 2015. – Т. 20. – No 1. – С. 75–103.

30. IntLinIncR2 – пакет программ для визуализации множеств решений интервальных линейных систем с двумя неизвестными. – Версия для MATLAB.: Релиз 01.09.2014. – URL: http://www.nsc.ru/interval/Programing/MCodes/IntLinIncR2_UTF8.zip.
31. Java Platform, Standard Edition (Java SE) 8. [Электронный ресурс]. – Заголовок с экрана. Режим доступа: www.docs.oracle.com/javase/8/index.html.
32. Overview (JavaFX 8). [Электронный ресурс]. – Заголовок с экрана. Режим доступа: http://docs.oracle.com/javase/8/javafx/api/overview-summary.html.
33. Knime Interval Tools [Электронный ресурс]. – Заголовок с экрана. Режим доступа: www.knimeintervaltools.java.net.
34. Eclipse Subversive - Subversion (SVN) Team Provider. [Электронный ресурс]. – Заголовок с экрана. Режим доступа: http://www.eclipse.org/subversive.
35. Шарый С. П., Шарая И.А. Распознавание разрешимости интервальных уравнений и его приложения к анализу данных // Вычислительные технологии. 2013. Т. 18, № 3. С. 80–109.
36. Калмыков С.А., Шокин Ю.И., Юлдашев З.Х. Методы интервального анализа. Новосибирск: Наука, 1986. – 223 с.
37. Hansen E. R. On linear algebraic equations with interval coefficients // Topics in Interval analysis, Oxford Press, 1969, pp. 35–46.
38. Kolev L. Interval Methods for Circuit Analysis. World Scientific Publishing Co. Pte. Ltd, 1993. 322 p.
39. Шипилов В. С. Программный модуль визуализации интервальных систем линейных алгебраических уравнений // Математика и ее приложения: фундаментальные проблемы науки и техники: сборник трудов всероссийской конференции, Барнаул, 24-26 ноября, 2015. – Барнаул: Изд-во Алт. ун-та, 2015. – 446 с.

ПРИЛОЖЕНИЕ

Исходный код узлов для интервального анализа в системе KNIME

Выпускная квалификационная работа выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

«___» _____ _____ г.

_____ _____

(подпись) *(Ф.И.О.)*