МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет математики и компьютерных наук**

**Кафедра вычислительной математики и информатики**

Допустить к защите

Заведующий кафедрой

д-р. тех. наук, доцент

Н. А. Наумова

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г.

Руководитель ООП

д-р. тех. наук, профессор

Ю. М. Вишняков

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г.

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**(МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)**

**ИНСТРУМЕНТЫ ДЛЯ РАЗРАБОТКИ СПЕЦИАЛИЗИРОВАННЫХ МАТЕМАТИЧЕСКИХ ПАКЕТОВ**

Работу выполнил                          Д. Т. Головатин

(подпись)

Направление подготовки 02.04.01 Математика и компьютерные науки .

Направленность (профиль) Вычислительная математика

Научный руководитель

канд. тех. наук, доцент   Е. Р. Алексеев

(подпись)

Нормоконтролер

преподаватель    А. А. Лахтина

(подпись)

Краснодар  
2025

**Содержание**

[Введение 4](#_Toc201058254)

[1 Обзор современных математических пакетов 6](#_Toc201058255)

[1.1 Пакеты общего назначения 7](#_Toc201058256)

[1.1.1 Wolfram Mathematica 7](#_Toc201058257)

[1.1.2 Scilab 9](#_Toc201058258)

[1.1.3 Sage 11](#_Toc201058259)

[1.1.4 MATLAB 13](#_Toc201058260)

[1.1.5 Mathcad 15](#_Toc201058261)

[1.2 Специализированные математические пакеты 17](#_Toc201058262)

[1.2.1 Язык R 17](#_Toc201058263)

[1.2.2 Электронные таблицы 18](#_Toc201058264)

[1.3 Языки программирования общего назначения как математические пакеты 20](#_Toc201058265)

[1.3.1 Python 20](#_Toc201058266)

[1.3.2 Julia 22](#_Toc201058267)

[1.3.3 Выводы 24](#_Toc201058268)

[2 Система открытой математики 25](#_Toc201058269)

[2.1 Базовая версия 25](#_Toc201058270)

[2.2 Расширенная версия 27](#_Toc201058271)

[2.3 Установка СОМ 28](#_Toc201058272)

[2.4 Система модулей СОМ 34](#_Toc201058273)

[2.5 Конфигурация через флаги 37](#_Toc201058274)

[3 Использование СОМ при решении задач 40](#_Toc201058275)

[3.1 Постановка задачи нахождения корней полинома 40](#_Toc201058276)

[3.1.1 Алгоритм парабол 40](#_Toc201058277)

[3.1.2 Модифицированный метод касательных 42](#_Toc201058278)

[3.1.3 Аппроксимация значения числа через ряды. 44](#_Toc201058279)

[3.2 Решение задачи нахождения корней полинома. 45](#_Toc201058280)

[3.2.1 Метод парабол 45](#_Toc201058281)

[3.2.2 Метод касательных 48](#_Toc201058282)

[3.2.3 Аппроксимация числа 49](#_Toc201058283)

[3.3 Результаты тестов СОМ 50](#_Toc201058284)

[3.3.1 Метод парабол 50](#_Toc201058285)

[3.3.2 Метод касательных 51](#_Toc201058286)

[3.3.3 Аппроксимация числа 52](#_Toc201058287)

[Заключение 55](#_Toc201058288)

[Список использованных источников 56](#_Toc201058289)

[Приложение А Краткое описание устанавливаемых пакетов 64](#_Toc201058290)

Введение

Математические пакеты — специализированные программные продукты, предназначенные для широкого класса задач. С каждым годом разрабатывается всё больше математического программного обеспечения с целью ускорения работы пользователя и минимизации вычислительных ошибок. Математические вычислительные пакеты играют важнейшую роль в науке и образовании: с развитием технологий математическое программное обеспечение позволило продвинуть науку на годы вперед, успешно справляясь с задачами, которые ранее человеку были непосильны ввиду их вычислительной сложности.

Благодаря существованию мощных программных продуктов для решения вычислительных задач, значительно упростилось изучение математики, физики, биологии и многих других естественных наук, так как современные инструменты визуализации и динамического моделирования позволяют досконально изучить исследуемые процессы, глубже понять их природу, за считанные минуты провести необходимые эксперименты.

Однако большинство современных математических пакетов — это сложные проприетарные инструменты, которые являются труднодоступными из-за высокой стоимости лицензии либо же из-за высоких требований к аппаратному обеспечению. Отсюда возникает необходимость в свободном математическом программном обеспечении, которое будет доступно рядовому пользователю.

В данной работе будет рассмотрен процесс построения свободного математического пакета.

Объектом исследования в данной работе является процесс построения специализированных математических пакетов.

Предмет исследования — специализированное программное обеспечение для построения математических пакетов с заданными свойствами.

Цель работы: разработка технологии построения модульных свободных специализированных математических пакетов.

1. Обзор современных математических пакетов

Появление цифровых технологий и экспоненциальный рост вычислительных мощностей существенно изменили процессы научной и образовательной деятельности. На стыке математики, информатики и прикладных дисциплин особую, если не центральную, роль стал играть специфический класс программного обеспечения — математические пакеты. Эти специализированные среды и библиотеки, предназначенные для выполнения широкого класса вычислений, моделирования и анализа данных, превратились из малоизвестных инструментов в неотъемлемый компонент исследовательского процесса и преподавания. В настоящий момент их обширный инструментарий позволяет решать задачи, ранее недоступные или требующие несоизмеримо больших временных и интеллектуальных затрат.

В современной науке математические пакеты стали незаменимым инструментом для решения вычислительных задач разной степени сложности.

Благодаря использованию математических пакетов исследователи избавились от необходимости заниматься рутинной, но объемной вычислительной работой. Использование математического программного обеспечения позволило снизить количество вычислительных ошибок при решении задач, таких как решение сложных систем дифференциальных уравнений, операции с матрицами большой размерности, обработка больших массивов экспериментальных данных, статистический анализ — это задачи, выполнение которых вручную практически невозможно. Теперь этим занимаются математические пакеты. Это позволяет ученым сосредоточиться на постановке задач, интерпретации результатов и построении новых теорий, а не на промежуточных громоздких выкладках.

Кроме того, вычислительные пакеты кардинальным образом изменили моделирование процессов реального мира и создание симуляций. Пакеты позволяют создавать сложные математические модели физических, биологических, химических, экономических и социальных систем, что раньше было невозможно. Теперь исследователи могут проводить различные эксперименты используя математическую модель, проверять гипотезы, прогнозировать поведение систем при изменении параметров, исследуя таким образом недоступные для прямого наблюдения процессы, например, в астрофизике или квантовой механике. Это значительно удешевляет и ускоряет проверку идей.

Благодаря инструментам визуализации, интегрированным в такие системы, теперь возможно представить абстрактные математические объекты и массивы данных в наглядной графической форме. Визуализация помогает глубже понять устройство исследуемого процесса, выявить скрытые закономерности и эффективно представить результаты исследования другим.

Не менее значима роль математических пакетов в образовательном процессе. Использование специализированного математического программного обеспечения позволяет сфокусировать внимание студентов на изучении математических концепций, пониманию логики решения, анализу и правильной интерпретации результатов. За счет возможности визуализировать сложные математические абстракции студенты легче осваивают материал, что способствует развитию математической интуиции.

* 1. Пакеты общего назначения
     1. Wolfram Mathematica

Wolfram Mathematica [1] — комплексная проприетарная система компьютерной математики общего назначения, цель которой объединить в себе различные области, использующие символьные и численные вычисления. Система состоит из языка Wolfram Language, который отвечает за решение вычислительных задач, обширной библиотеки интегрированных алгоритмов, предоставляющих пользователю широкие возможности для научной и инженерной деятельности, а также интегрированного интерфейса Notebook, совмещающего в себе возможности для написания кода, вставки различного мультимедиа контента и интерактивных элементов.

Wolfram Language — символьный язык программирования, который является ядром экосистемы Wolfram. Работа на этом языке осуществляется через написание специальных выражений. Через выражения пользователь описывает что он хочет получить в результате, а не пошаговый алгоритм действий, как это принято в императивных языках. В проекте реализована система правил и сопоставлений, благодаря которым упрощается процесс определения математических абстракций. С ее помощью можно задавать правила упрощения, преобразования, определять собственные математические конструкции. Можно задавать правила упрощения, преобразования, определять собственные конструкции. За счет этого Mathematica является одним из самых мощных инструментов для работы с символьными вычислениями. Помимо этого, в языке реализована богатая система типов данных и структур, среди которых присутствуют целые числа произвольной длины, рациональные числа, иррациональные числа, вещественные числа произвольной точности, разного рода матрицы и многие другие. Кроме того, поддерживается работа со специализированными математическими символами [2]. Также поддерживается работа с параллельными и распределенными вычислениями [3], позволяющая писать производительный код.

В Wolfram Mathematica реализована богатая библиотека методов, позволяющая решать как классические инженерные и вычислительные задачи, так и использовать систему в биологии [4], медицине и экономике [5].

Базовая функциональность Mathematica возможно расширить через установку официальных дополнений, которые добавляют поддержку специализированных инструментов, либо же через модули Paclets, которые пользователи могут разрабатывать и распространять самостоятельно.

Помимо платного проприетарного решения Wolfram Mathematica, пользователи бесплатно могут установить себе Wolfram Engine и работать с системой через консольный интерфейс, либо же через Jupyter Notebook после дополнительной настройки. В Wolfram Engine пользователь получает возможность Слабыми сторонами системы являются ограничивающая возможности пользователя проприетарная лицензия с высокой стоимостью и большой объем занимаемого места на компьютере. Первый фактор не позволяет исправлять ошибки в алгоритмах, вынуждая ждать обновление со стороны разработчиков. А стоимость лицензии может отпугнуть рядового пользователя и осложняет использование системы в образовательных учреждениях. А второй фактор может помешать пользоваться системой на слабом компьютере, потому как размер установленного дистрибутива Wolfram Mathematica вместе с документацией может достигать 18 гигабайт. Стоит также отметить, что многим пользователям может потребоваться значительное время на освоение системы, так как использование выражений вместо классических императивного и объектно-ориентированного подходов требует определенного осмысления и привыкания.

* + 1. Scilab

Scilab [6] — свободный математический пакет, ориентированный на решение численных задач. Работа в Scilab осуществляется через написание программ на встроенном языке высокого уровня, который синтаксически очень схож с MATLAB [7]. В пакете реализовано множество алгоритмов для решения разного рода вычислительных задач, таких как задачи линейной алгебры, задачи численного интегрирования и дифференцирования, аппроксимации данных. Для численных расчетов в Scilab используются библиотеки Lapack, LINPACK, ODEPACK, Atlas и другие, на основе которых работает большинство современного вычислительного программного обеспечения. Кроме того, пакет оснащен мощными средствами визуализации, доступно построение 2D и 3D графиков с тонкой настройкой легенды. Полученную визуализацию можно экспортировать в растровые и векторные форматы. Имеется поддержка интегрированной среды динамического моделирования Xcos, данная среда является аналогом Simulink. В Xcos реализована обширная библиотека блоков, среди которых присутствуют математические операции, различные источники сигналов, электрические компоненты, системы управления и многое другое [8]. Также поддерживается вызов программ написанных на C, C++ и Fortran. За счет описанных факторов Scilab стал активно используется учеными [9], которые хотят простой, но мощный инструмент для решения вычислительных задач.

Возможности Scilab можно расширить благодаря ATOMS [10]. ATOMS — система управления внешними зависимостями, которая позволяет пользователю устанавливать сторонние модули из экосистемы Scilab. Благодаря ей упрощается процесс установки и настройки зависимостей. В репозиториях ATOMS пользователь может найти расширения, добавляющие в Scilab возможности обработки видео и изображений, расширенные методы обработки сигналов, дополнительные алгоритмы для задач оптимизации, модули для взаимодействия с Arduino и Rasberry Pi и многое другое. При желании пользователь может написать собственный модуль.

Благодаря богатому функционалу и свободной лицензии Scilab активно используется в образовательных учреждениях [11-13]. Пакет занимает небольшой объем пространства на диске, около 600 мегабайт, что намного меньше большинства проприетарных аналогов, таким образом, система может быть установлена на компьютер с ограниченными ресурсами. На факультете математики и компьютерных наук Scilab используется в курсе технологии программирования и работы на ЭВМ, а также при обучении педагогов на курсе свободных математических пакетов.

Также существует ядро Scilab для Jupyter, это позволяет сделать из Scilab интерактивную среду, где пользователю сможет не только проводить вычисления, но и писать лекции, вести отчеты.

К особенностям Scilab можно отнести низкую скорость работы в некоторых вычислительных задачах, относительно небольшое сообщество, если сравнивать с MATLAB и Python.

* + 1. Sage

Свободная система компьютерной алгебры Sage [14] написанная на Python, задумывалась как альтернатива проприетарным системам, таким как MATLAB, Wolfram Mathematica, Maple и Magma. Основная идея Sage — создать математический пакет, использующий свободные вычислительные инструменты для решения исследовательских и научных задач. Sage — это дистрибутив, в который включены различные дополнительные компоненты. Sage состоит из языка программирования Python, на котором написано ядро вычислительной системы, реализованы собственные классы, обеспечивающие удобство при работе с математическими объектами. В Sage реализована система автоматического приведения типов, которая позволяет конвертировать Python типы в типы Sage автоматически. Компоненты системы делятся на обязательные, которые входят в ядро среды, а также на опциональные которые могут быть установлены отдельно через встроенный инструмент управления пакетами. Помимо этого, благодаря использованию Python система поддерживает расширение через установку сторонних библиотек из PYPI. В пакет интегрировано множество специализированных математических инструментов. Некоторые из доступных модулей с разделением на категории решаемых задач приведены ниже.

1. Символьные вычисления и вычислительная алгебра: SymPy, Maxima, GAP, Singular.
2. Численные вычисления: NumPy, SciPy, GSL, ATLAS, OpenBLAS;
3. Теория чисел: PARI/GR, NTL, FLINT.
4. Визуализация данных: Matplotlib.

Кроме этого, для решения статистических задач Sage использует язык программирования R. Полный список используемых модулей представлен в официальной документации [15]. Также присутствует поддержка параллельных вычислений. Также присутствует возможность для генерации скомпилированного кода посредством Cython [16] — статического компилятора Python. За счет этого Sage является мощным инструментом, способным решать задачи из разных разделов математики и физики.

Работа в Sage осуществляется либо через интерпретатор в терминале, либо через Jupyter. Второй способ предпочтительнее, так как открывает возможности для создания интерактивных блокнотов, схожих по функциональности с Wolfram Mathematica.

Из особенностей системы можно отметить, во-первых, сложную установку, так как система содержится не во всех репозиториях Linux, поэтому в некоторых версиях дистрибутивов пользователю придется собирать систему из исходников самостоятельно. Во-вторых, Sage занимает внушительный объем пространства на диске, так как является комплексной системой со множеством внешних компонентов, интегрированных в среду. У пользователя нет полного контроля над используемыми модулями, он может управлять только опциональными библиотеками. Кроме того, так как Sage использует свободные вычислительные средства для решения задач, они могут быть не так удобны в использовании и не так производительны в сравнении с проприетарными аналогами.

Тем не менее, Sage активно используется в научной деятельности при решении задач алгебры [17]−[18] и в обучении.

* + 1. MATLAB

MATLAB [19] (MATrix LABoratory) — это проприетарная интерактивная среда для численных вычислений с богатым функционалом для решения разного рода задач. В системе поддерживается программирование на собственном языке высокого уровня, который легко осваивается учеными и студентами технических специальностей. Язык поддерживает процедурное, функциональное и объектно-ориентированное программирование, пользователи могут создавать как скрипты, так и полноценные приложения. Язык удобен для прототипирования, что позволяет быстро набросать алгоритм, который впоследствии можно переписать на более производительных языках, таких как C или C++.

В системе также реализованы широкие возможности для визуализации данных, поддерживается построение гистограмм, диаграмм рассеивания, контурных графиков, поверхностей и других.

Одной из преимуществ среды является — поддержка наборов инструментов (Toolboxes, Тулбоксы). Наборы инструментов — это специализированные надстройки, расширяющие функциональность системы. Известными тулбоксами являются:

1. Signal Processing Toolbox — инструменты для фильтрации и обработки сигналов;
2. Image Processing Toolbox — предоставляет возможности для обработки изображений;
3. Optimization Toolbox — предоставляет возможности для решения задач линейной, численной оптимизации;
4. Statistic and Machine Learning Toolbox — содержит алгоритмы для решения задач статистического анализа и машинного обучения;
5. Deep Learning Toolbox — предоставляет инструменты для проектирования, обучения и развертывания глубоких нейронных сетей;
6. Simulink — среда графического моделирования и симуляции динамических систем;
7. и многие другие, с полным списком которых можно ознакомиться на официальном сайте [20].

Тулбоксы не входят в базовую лицензию MATLAB, лицензию на необходимые тулбоксы нужно приобретать отдельно.

Также в системе поддерживается возможность создания приложений с графическим интерфейсом благодаря App Designer.

Поддерживается интеграция с языками C и C++, Fortran и Python, благодаря которой возможно как использование программ на вышеописанных языках, так и вызов функций MATLAB внутри программ на сторонних языках.

За счет простого синтаксиса и мощных инструментов MATLAB обрел популярность в научной среде [21]. В академической среде MATLAB тоже используется для наглядного представления [22] математических идей и концепций. Простой язык среды быстро осваивается студентами, что позволяет студентам сосредоточиться на изучении математики, а не на синтаксических конструкциях.

Основные особенности MATLAB — закрытость с высокой стоимостью лицензии; пользователю потребуется купить не только сам MATLAB, но и докупить необходимые тулбоксы, что мешает распространению продукта среди индивидуальных пользователей и небольших образовательных организаций. Существуют различные типы лицензий, в том числе академическая, что упрощает приобретение продукта, однако она ограничена набором доступных инструментов.

Кроме проблем с лицензией, весомым негативным фактором для некоторых пользователей является размер занимаемого пространства на диске. Базовая версия системы занимает около 4−5 гб, однако размер полной версии со установленными популярными тулбоксами может превысить 20 гб.

При решении некоторых особо тяжелых вычислительных задач производительность системы может отставать от конкурентов, а также от аналогичных программ, написанных в языках C и C++.

* + 1. Mathcad

MathCad [23] — проприетарная система компьютерной алгебры, ориентированная на подготовку интерактивных документов состоящих из вложенных вычислений и визуального представления данных. Главным отличием MathCad от большинства математических пакетов является визуальный интерфейс типа WYSIWYG (What You See Is What You Get — «Что видишь, то и получаешь») — эта особенность позволяет вводить математические формулы и отображать их так, как если бы они были написаны руками на листе бумаги. За счет этого MathCad обрел большую популярность в образовательной среде [24], так как избавляет пользователя от необходимости изучать синтаксис языков программирования, используемых в других системах как основных инструменты взаимодействия со средой.

В системе реализован широкий инструментарий для решения различных вычислительных задач, система способна:

1. численно решать алгебраические уравнения и системы уравнений, как линейные, так и нелинейные;
2. проводить различные операции с векторами и матрицами;
3. решать задачи численного дифференцирования и интегрирования;
4. решать задачи оптимизации, статистического анализа и многие другие.

В системе присутствует движок для выполнения символьных вычислений, благодаря которому система способна:

1. упрощать выражения;
2. проводить аналитическое дифференцирование и интегрирование;
3. решать уравнения и системы уравнений в символьном виде;
4. проводить различные преобразования такие как, например, преобразование Лапласа и преобразование Фурье;
5. проводить различные операции с полиномами.

В программе также поддерживается возможность построения различных 2D и 3D графиков с гибкой настройкой обозначений, а также создание анимаций.

Еще одна особенность MathCad — встроенная система для работы с физическими величинами и их единицами измерения. Пакет автоматически отслеживает корректность размерностей и выполняет преобразования единиц. Это значительно снижает вероятность ошибок в инженерных расчетах [25].

В проекте реализованы элементы процедурного программирования такие как циклы, поддержка пользовательские функций, условный оператор — что позволяет разрабатывать сложные вычислительные алгоритмы.

Благодаря работе в среде типа WYSIWYG проведенные расчеты, графики и формулы сохраняются в едином документе и если пользователь изменит какие-либо данные, то все зависимые расчеты и графики автоматически обновляться. Это делает MathCad удобным инструментов для создания различных отчетов [26].

Также в MathCad оснащен богатой справочной информацией [27] в формате интерактивных документов, что позволяет быстро освоить возможности системы на наглядных примерах.

Несмотря на удобство использования среды за счет наглядности выполняемых операций, система имеем ряд проблем.

Во-первых, в системе отсутствуют гибкие возможности для расширения функциональности. Поддерживается только интеграция с другими продуктами компании, либо подключение сторонних С++ программ, что вынуждает пользователя заниматься поиском и настройкой соответствующих C++ библиотек самостоятельно. Так же присутствует интеграция с Microsoft Excel, что может быть полезно при решении определенных задач. Ранее система поддерживала пакеты расширений, схожие по принципу с MATLAB Toolboxes, однако в последний версиях данные расширения либо интегрированы в систему, либо их поддержка прекращена.

Во-вторых, проприетарная лицензия ограничивает свободу пользователя и может послужить барьером для использования системы в ряде организаций.

В-третьих, вычислительные возможности Mathcad в ряде задач уступают таким популярным вычислительным средам как MATLAB и Wolfram Mathematica, из-за чего использование среды при решении подобных задач оказывается невозможным.

* 1. Специализированные математические пакеты
     1. Язык R

Язык R [28] — свободный объектно-ориентированный язык программирования, разработанный для решения статистических задач. В языке реализовано множество функций для проведения статистического анализа и визуализации данных, язык способен покрыть все задачи статистического анализа [29] за счет чего он активно используется в научной среде [30-31]. Благодаря специализации на конкретной области — статистическом анализе, процесс решения описанных задач в R является более простым, чем в инструментах общего назначения. Кроме того, поддерживается параллельная обработка данных. Существуют различные среды с поддержкой данного языка, такие как RGui, которая поставляется вместе с языком, а также Rstudio — интегрированная среда разработки, ориентированная на работу с R. Она устанавливается отдельно.

Язык оснащен мощными инструменты визуализации, библиотека ggplot2 позволяет выводить не только различные виды диаграмм, но также и различные картографические построения. Благодаря этому R активно используется при решении задач геофизики [32].

R из коробки поддерживает широкие возможности для импорта и экспорта данных, поддерживаются такие популярные форматы файлов как: xml, csv, tsv и другие. Поддерживается и работа с двоичными табличными форматами популярных программ: Excel, SPSS, SAS, Stata.

Экосистема R насчитывает более 22000 пакетов для анализа данных в своем репозитории [33]. Пакеты в экосистеме R оснащены подробной документацией, которая способствует быстрому освоению языка, кроме того, многие ключевые пакеты для анализа данных в R предоставляют не только функции для обработки данных, но и сами наборы данных. Некоторые пакеты не имеют аналогов в других вычислительных системах. Существует рецензируемый научный журнал — «The R Journal» [35], в котором публикуются открытые исследования в области статистического анализа. В нем читатели могут свободно ознакомиться не только с полными текстами статей, но и с их исходными кодами. В журнале так же освещаются новые библиотеки экосистемы R.

Среди особенностей R можно отметить: нестандартный синтаксис, из-за которого могут возникнуть сложности при изучении языка, низкая производительность при решении некоторых задач, так как R является интерпретируемым языком.

* + 1. Электронные таблицы

В рамках анализа современных математических пакетов особое место занимают табличные процессоры, такие как Microsoft Excel и Google Spreadsheets. Несмотря на то, что их основное предназначение — офисная работа с табличными данными, их функциональность позволяет решать широкий круг вычислительных, статистических и оптимизационных задач. Благодаря своей повсеместной распространенности и низкому порогу вхождения, эти инструменты часто используются среди студентов, инженеров и даже ученых.

Excel и Spreadsheets — это проприетарные программы, предназначенные для обработки табличных данных. Пакеты не являются программным обеспечением, ориентированным на решение вычислительных задач, тем не менее эти инструменты могут быть использованы для решения базовых задач численного и статистического анализа. Например, в системах реализованы функции для численного решения простых задач линейной алгебры, таких как решение систем линейных уравнений, поиск обратной матрицы, нахождение определителя. Решение статистических задач в Excel осуществляется через множество встроенных функций, таких как: «СРЗНАЧ», «СТАНДОТКЛОН», «СТЬЮДРАСПОБР» и других. Для Spreadsheets существует аналогичный набор функций. Кроме того, в Excel поддерживаются надстройки «Поиск решения», и «Анализ данных», которые позволяют решать задачи динамического программирования и некоторые сложные задачи статистического анализа. В Spreadsheets подобные задачи решаются через инструмент «Solver».

В системах реализованы широкие возможности визуализации данных, поддерживаются различные диаграммы и гистограммы.

Кроме того, в Excel поддерживается скриптовой язык VBA (Visual Basic for Application), который позволяет разрабатывать сложные алгоритмы обработки табличных данных.

Системы часто используются студентами для несложного анализа и предобработки данных, так как эти программы проще в освоении нежели более специализированное программное обеспечение. Главными преимуществами программ являются их распространенность, каждый хоть немного, но знаком с Excel или аналогичной программой, а также относительная простота использования этих средств, в них просто освоиться, это быстрее чем изучение специализированного программного обеспечения.

К особенностям средств можно отнести проприетарную лицензию, а также технические ограничения, не позволяющие решать сложные вычислительные задачи большой размерности.

* 1. Языки программирования общего назначения как математические пакеты

Такие языки программирования, как Python и Julia являются популярными инструментами в исследовательской и инженерной среде, поэтому рассмотрим их преимущества и проблемы.

* + 1. Python

Python [35] — это высокоуровневый динамический язык программирования общего назначения с открытой лицензией, поддерживающий различные парадигмы программирования, такие как процедурное, структурное, объектно-ориентированное, функциональное и др.

На сегодняшний день это один из самых популярных языков программирования. Благодаря простому синтаксису, язык легко осваивается как студентами [36]−[37], так и исследователями [38]. За счет этого Python стал основным языком для быстрого прототипирования.

Язык имеет поддержку менеджера пакетов, благодаря чему пользователь может расширить функционал языка. Экосистема Python содержит большое количество библиотек разной направленности. Главный репозиторий PyPI насчитывает более 600 тысяч различных пакетов [39] и этот список постоянно пополняется. В экосистеме Python можно найти библиотеки для решения практически любых задач.

Python изначально не задумывался как язык для решения научных задач, тем не менее, за счет простого синтаксиса и пакетов NumPy, SciPy, Matplotlib, завоевал признание среди ученых. NumPy — библиотека, написанная на C для обработки многомерных массивов. В ней реализованы основные алгоритмы для работы с матрицами, а за счет использования векторизованных операций обеспечивается высокая скорость выполнения операций при использовании библиотеки. На базе данного модуля построено разнообразное множество вычислительных пакетов для анализа данных, эффективной обработки больших данных, машинного обучения [40]. SciPy — пакет, использующий NumPy, в котором реализованы методы для решения задач оптимизации, интерполяции и статистики. Он активно используется физиками и математиками при решении задач линейной алгебры, обыкновенных дифференциальных уравнений и многих других. Matplotib — библиотека для построения 2D и 3D визуализаций. Она позволяет создать как статические, так и динамические графики с поддержкой интерактивного взаимодействия, дополняя возможности языка Python для решения научных задач. Описанные библиотеки являются самыми известными инструментами для научной разработки, однако в экосистеме языка достаточно и узкоспециализированных модулей под конкретные области и задачи, таким образом пользователь гарантировано найдет нужный модуль для решения необходимых задач.

При всех преимуществах язык не лишен проблем. К основным недостаткам языка относят скорость его работы, программы на Python исполняются медленнее, чем аналогичные на C. Эта проблема отчасти решается использованием библиотек, написанных на более производительных языках, как, например, упомянутый выше NumPy который написан на C, однако при использовании родных механизмов языка, низкая скорость выполнения будет заметна и это сказывается на всей программе в целом. Кроме того, в Python существует проблема несовместимости библиотек. Часто возникает ситуация, когда в системе есть несколько фреймворков, которые требуют разные версии одной и той же библиотеки. В настоящий момент эта проблема решается использованием изолированных виртуальных окружений, либо же, сторонними менеджерами пакетов, которые занимаются управлением окружениями. Кроме того, до недавнего времени в Python не было поддержки параллельных потоков, Это связано с глобальной блокировкой интерпретатора, однако в современном Python (3.13+) ведется работа по решению данной проблемы.

* + 1. Julia

­­

Julia [41] — Свободный высокоуровневый язык программирования общего назначения, созданный для решения вычислительных задач. Разработчики Julia стремились создать мощную свободную альтернативу проприетарным математическим пакетам. Синтаксис Julia вдохновлен синтаксисом Matlab, но в то же время похож на Python, за счет чего язык осваивается быстро как студентами, так и учеными. В Julia реализована гибкая система типов, способствующая разработке математических алгоритмов, поддерживаются массивы, матрицы, рациональные числа, длинные целые и вещественная арифметика заданной точности. Так же поддерживается как создание абстрактных типов, так и конкретных, за счет этого пользователи могут строить гибкую иерархию типов внутри своих программ. Главной особенностью Julia является использование JIT [42] (Just-in-time) компилятора, что в совокупности с динамической типизацией позволяет как заниматься быстрым прототипированием по аналогии с Python, так и генерировать высокопроизводительный код благодаря неполной компиляции. Пользователь путем аннотирования типов внутри функций и структур может указать компилятору как корректно скомпилировать этот фрагмент кода, за счет чего на выходе получается высокопроизводительный код, сравнимый по скорости с Fortran и C++ [43]. Достоинством языка так же является использование множественной диспетчеризации [44]. Благодаря этому пользователи с легкостью могут расширять функциональность встроенных пакетов и функций, что дает широкие возможности в плане гибкости и расширяемости.

В Julia реализован удобный пакетный менеджер, который позволяет устанавливать и настраивать различные опциональные библиотеки из общего репозитория [45]. Экосистема языка не так обширна, как у Python, однако экосистема Julia в большей степени ориентирована на научные вычисления и решение инженерных задач, нежели экосистема Python. В репозитории представлено большое количество модулей, ориентированных на решение узкоспециализированных задач, таких как, например, решение гидростатических уравнений [46] и многих других специфических задач. Помимо узкоспециализированных пакетов, присутствуют пакеты, представляющие собой полноценные самостоятельные системы для решения задач динамического моделирования [47] и символьной алгебры [48].

Кроме того, язык богат на инструменты для выполнения параллельных вычислений, программирования на GPU.

Тем не менее, кроме преимуществ, в языке присутствует ряд проблем. Во-первых, на сегодняшний момент не существует полноценной среды разработки для Julia. Единственная близкая по функционалу альтернатива – Visual Studio code с установленным плагином для Julia. Данный плагин обеспечивает широкие возможности для взаимодействия с языком отладку, профилирование, автодополнение кода и многое другое, однако VS code является универсальным редактором кода, который ввиду своей всеядности проигрывает по производительности и функциональности специализированным средам разработки.

Во-вторых, у Julia небольшое сообщество, поэтому экосистема развивается медленно. Многие пакеты или заброшены или обновляются очень редко, поэтому поиск подходящей библиотеки под задачи конкретного пользователя может вызвать трудности. Так же стоит упомянуть, что Julia сложно конкурировать с Python в области машинного и глубокого обучения. Фреймворки Julia отстают от возможностей конкурентов.

Однако несмотря на озвученные проблемы, Julia завоевала некоторую популярность в научной среде [49] и активно используется на факультете математики и компьютерных наук [50]−[51].

* + 1. Выводы

Современные математические пакеты обладают широкими возможностями для решения инженерных и вычислительных задач, однако у каждого есть особенности. Главными проблемами популярных математических пакетов являются проприетарность и большой размер установленной среды. При этом зачастую пользователю не нужен весь функционал пакета, а лишь отдельные его функции, но пользователь будет вынужден устанавливать пакет целиком. Необходима возможность гибкой настройки системы с возможностью выбора нужного функционала.

Поэтому было принято решение разработать математическую среду на основе свободных технологий, которую пользователь сможет настроить под себя, установив только те вычислительные ядра, которые ему необходимы. Это решит проблему большого дистрибутива и позволит создать специализированную вычислительную среду под задачи конкретного пользователя. Для этого система должна быть модульной и расширяемой. Также необходимо сделать установку системы простой и интуитивной, чтобы даже неопытный пользователь смог установить и настроить среду.

1. Система открытой математики

«Система открытой математики» (СОМ) — cвободный математический пакет, разрабатываемый для операционных систем на базе Linux, поддерживающий работу с различными вычислительными ядрами, а также имеющий систему модулей, которые расширяют базовый функционал языков Julia и Python. СОМ предоставляет пользователю готовую среду для решения вычислительных задач, написания интерактивных отчетов [52], ведения инженерных конспектов, а также дает возможность создавать специализированные среды под нужны конкретного пользователя.

* 1. Базовая версия

Разработка СОМ началась в рамках бакалаврской работы [53] исходя из стремления достичь цели, описанные выше. По результатам работы в разработанной версии проекта был реализован следующий функционал:

1. автоматическая установка веб-среды Jupyter;
2. автоматическая установка и настройка языка Julia и вычислительного ядра IJulia для Jupyter;
3. возможность выбора версии Julia для установки;
4. установка и настройка модулей для Julia.

Помимо этого, было разработано русскоязычное руководство по языку программирования Julia, представляющее собой курс по основам синтаксиса языка. Целью данного курса было упростить знакомство с языком и его экосистемой.

Данная версия состояла из установочного скрипта *installer.sh*, написанному на bash [54]−[55] благодаря которому осуществлялась установка основных компонентов СОМ: среды Jupyter, языка Julia и его модуля *IJulia*.

Jupyter был выбран в качестве основной среды для программирования, так как данный инструмент предоставляет наиболее богатые возможности для решения вычислительных задач, поддерживает вывод графики, таблиц, мультимедиа, использование математических символов как в *TeX* и является продуктом со свободной лицензией.

В качестве главного языка системы была выбрана Julia, так как по описанным выше причинам является самым близким по функциональности свободным вычислительным движком по отношению к проприетарным альтернативам.

Для подключения Julia в Jupyter используется специальное ядро *IJulia*, за счет него обеспечивается работоспособность Julia в среде.

Для достижения простоты при установке СОМ процесс был реализован в терминале в виде серии вопросов и ответов, где пользователю необходимо ответить согласен ли он с данными параметрами или нет. После установки основных компонентов среды, установочный скрипт запускал следующий этап установки — установку модулей.

Установка модулей осуществлялась при помощи скрипта:

*package\_installer.jl*

В данной программе при помощи интерактивного терминального меню пользователь мог выбрать, какие библиотеки установить из заранее отобранные пакетов, которые были разделены на группы (наборы) по задачам, которые они решают в рамках СОМ. В состав данных наборов вошли самые популярные пакеты экосистемы, которые охватывают основные задачи, с которыми сталкивается студент при обучении на техническом факультете, а также задачи, с которыми чаще всего сталкивается исследователь, либо ученный в процессе работы.

Достоинством реализованной системы модулей является возможность модификации указанного списка модулей, что позволяет преподавателю, либо же любому желающему изменить список, добавляя или удаляя модули по своему желанию. Если изначальный список будет изменен, то при повторном запуске *package\_installer.jl* пользователю будет доступен уже модифицированный список модулей для установки. Это позволяет СОМ создавать уникальные среды под конкретные задачи.

* 1. Расширенная версия

В рамках магистерской диссертации была разработана обновленная версия проекта, были добавлены следующие нововведения: переписан процесс установки вычислительных ядер для Jupyter, добавлена поддержка JupyterLab в дополнение к Jupyter Notebook, реализованы алгоритмы установки таких свободных вычислительных ядер как Scilab, Octave, Python, Bash. Для включения ядра Scilab в СОМ была написана функция *scilab\_kernel\_install*. Аналогично установка Octave реализована через функцию *octave\_kernel\_install*. Благодаря поддержке Scilab и Octave в СОМ, пользователи смогут разрабатывать математические документы на знакомом им языке, имея при этом возможность пользоваться всеми существующими наработками, а также всеми возможностями Scilab и Octave. Кроме того, добавлена поддержка вычислительного ядра Wolfram Engine. Реализована соответствующая функция. Ядро Wolfram Engine не является свободным, но является бесплатным и достаточно мощным, поэтому если пользователь захочет, может настроить и его в том числе. Несмотря на то, что поддержка языка программирования Python присутствует в Jupyter изначально, СОМ расширяет возможности Python, добавляя в среду специализированные наборы библиотек, которые отсутствуют в ядре по умолчанию. Поддержка ядра Bash была добавлена, так как язык Bash используется в курсе изучения ОС Linux и является основным инструментом для управления системой из терминала. Знакомство с языком необходимо для углубления познаний студентов о Linux. Русскоязычное руководство было заменено курсом лекций, написанным для педагогического направления факультета математики и Компьютерных наук КубГУ. Проект, как и раньше состоит из двух скриптов, которые отвечают за установку и настройку: *installer.sh* и *package\_installer.jl.*

Рассмотрим процесс установки СОМ поподробнее.

* 1. Установка СОМ

Пользователю необходимо загрузить *installer.sh* и *package\_installer.jl* с репозитория [56]. Затем, нужно выполнить в терминале следующий набор команд.

1. Предоставить installer.sh права на исполнение, прописав

sudo chmod +x installer.sh

1. Запустить программу выполнив команду ./installer.sh

После запуска скрипта пользователю будет выведено приветственное сообщение (рисунок 1), в котором описываются текущие настройки установщика. Пользователя попросят ввести пароль root, затем будет предложено продолжить установку с данными настройками. Также будет выведена справочная информация, в которой демонстрируются поддерживаемые флаги установки, подробнее о флагах будет рассказано дальше.

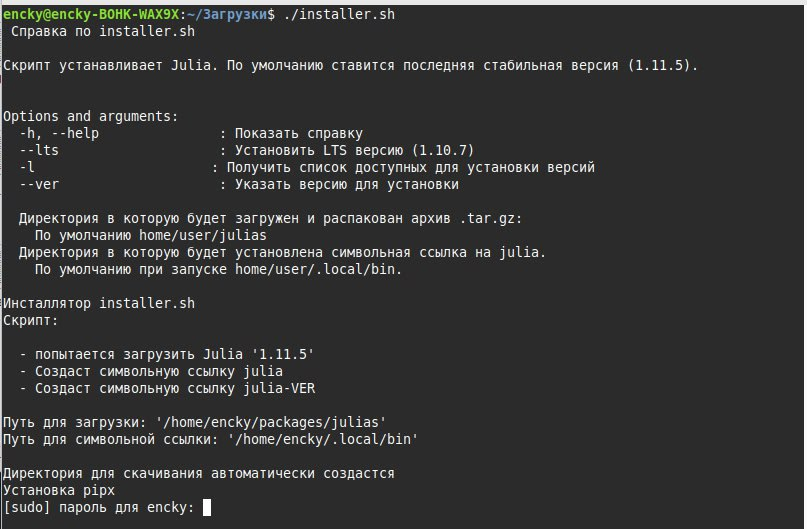


Рисунок 1 – Приветственное сообщение

Установка и настройка СОМ происходит поэтапно через серию вопросов и ответов. Пользователь должен будет ответить «Y» или «Д» — если он согласен запустить данный шаг установки, либо «N» или «Н» — если он отказывается. Интерпретируются ответы, введенные как заглавными буквами, так и строчными. Всего имеется 7 этапов:

1. загрузка и установка Jupyter;
2. загрузка и установка Julia;
3. загрузка и установка Scilab kernel;
4. загрузка и установка Bash kernel;
5. загрузка и установка Octave kernel;
6. загрузка и установка Wolfram kernel;
7. установка модулей для Julia и Python;

Во время первого этапа осуществляется установка Jupyter, причем в зависимости от установленной версии языка Python в системе пользователя, будет различаться и метод установки. Для систем с Python версии 3.11 и выше установка будет осуществляться через pipx, а для систем с версиями ниже — через pip. Данная особенность обусловлена тем, что СОМ устанавливает Jupyter и его зависимости таким образом, чтобы они были доступны глобально. Это упрощает процесс использования системы для неопытных пользователей, однако в операционных системах Linux начиная с версии 3.11 и выше установка Python приложений в глобальное окружение была запрещена, так как есть риск конфликта версий со встроенными python зависимостями, на которых работает операционная система. Чтобы избежать возможных проблем с операционной системой и при этом сохранить у СОМ свойство быть доступным отовсюду, было принято решение адаптировать процесс установки под использование pipx.

Данный шаг полностью автоматический, алгоритм сам выберет как именно установить Jupyter. Функция *check\_python\_version* проверяет версию Python в системе и изменяет значение переменной *use\_pipx,* которая отвечает за переключение режима установки.

function check\_python\_version() {

if [[ "$(printf '%s\n' "3.11" "$PYTHON\_VERSION" | sort -V | tail -n 1)" == "$PYTHON\_VERSION" ]]; then

USE\_PIPX=1

install\_pipx

fi

}

Если системе используется Python 3.11+, то в систему сначала устанавливается сам pipx через вызов *install\_pipx*, а затем Jupyter.

function install\_pipx(){

echo "Установка pipx"

sudo apt install pipx

pipx ensurepath

source ~/.bashrc}

function jupyter\_install() {

if [[ "$JUPYTER\_INSTALLED" -eq 0 ]]; then

echo "Установка jupyter"

if [[ "$USE\_PIPX" -eq 1 ]]; then

pipx install --include-deps jupyter

else

python3 -m pip install notebook

fi

fi

}

На втором этапе пользователю будет предложено начать установку Julia (рисунок 2). По умолчанию загружается последняя версия языка. Информация о том, какая версия языка в настоящий момент является последней, хранится в переменной *JULIA\_LATEST.* Значение данной переменой конфигурируется вручную разработчиком СОМ, однако оно может быть изменено пользователем, если он того пожелает.

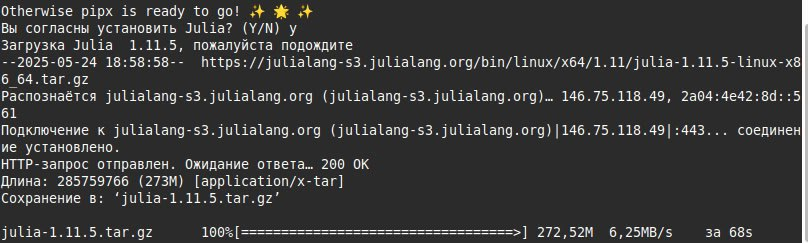


Рисунок 2 – Скриншот процесса установки Julia

Язык устанавливается в следующую директорию *home/user/.local/bin* и будет доступен по команде julia*.*

Алгоритмом проверяется архитектура процессора пользователя, затем загружается архив с исполняемыми файлами языка, происходит распаковка архива, копирование содержимого в указанную директорию и создание символьной ссылки. Также предусмотрен сценарий, если у пользователя язык уже установлен, в таком случае ему будет выведено соответствующее сообщение, а данный шаг будет пропущен.

Третьим шагом выполняется установка ядра Scilab для Jupyter, на данном шаге, а также во время установки ядер Octave и Bash предполагается что у пользователя установлены соответствующие дистрибутивы математических пакетов, если их нет, то при попытке продолжить установку ядер, система автоматически установит необходимые среды.

function scilab\_kernel\_install(){

read -rp "Хотите установить Scilab kernel в Jupyter? (y/n): " reply

if [[ "$reply" =~ ^[YyДд]$ ]]; then

# Проверка наличия scilab

if ! command -v scilab &> /dev/null; then

sudo apt install scilab

fi

echo "Продолжаем установку ядра..."

if [[ "$USE\_PIPX" == "1" ]]; then

echo "Установка через pipx..."

pipx inject --include-apps --include-deps jupyter scilab\_kernel

PIPX\_PATH="$(pipx environment | grep PIPX\_LOCAL\_VENVS | cut -d= -f2)"

source "$PIPX\_PATH"/jupyter/bin/activate

python3 -m scilab\_kernel install --user

deactivate

else

echo "Установка через pip..."

pip install --user scilab\_kernel

python3 -m scilab\_kernel install --user

fi

echo "Ядро Scilab успешно добавлено в Jupyter!"

return 0

fi

}

В данной функции в зависимости от версии Python установка осуществляется либо в глобальное окружение, либо в изолированное. Изолированное окружение создано через pipx и оно позволяет избежать проблем с зависимостями, однако это усложняет процесс активации ядра в Jupyter, так как требует дополнительных действий по настройке.

Четвертый и пятый шаги аналогичны предыдущему, меняется только названия ядер. Функции представлены ниже.

Функция *bash\_kernel\_install,* отвечающая за установку ядра Bash в Jupyter:

function bash\_kernel\_install(){

read -rp "Хотите установить Bash kernel в Jupyter? (y/n): " reply

if [[ "$reply" =~ ^[Yy]$ ]]; then

if [[ "$USE\_PIPX" == "1" ]]; then

echo "Установка через pipx..."

pipx inject --include-apps --include-deps jupyter bash\_kernel

PIPX\_PATH="$(pipx environment | grep PIPX\_LOCAL\_VENVS | cut -d= -f2)"

source "$PIPX\_PATH"/jupyter/bin/activate

python3 -m bash\_kernel.install --user

deactivate

else

echo "Установка через pip..."

pip install --user bash\_kernel

python3 -m bash\_kernel.install --user

fi

echo "Ядро Bash успешно добавлено в Jupyter!"

return 0

fi

}

Функция *octave\_kernel\_install,* отвечающая за установку ядра Octave в Jupyter:

function octave\_kernel\_install(){

read -rp "Хотите установить Octave kernel в Jupyter? (y/n): " reply

if [[ "$reply" =~ ^[Yy]$ ]]; then

# Проверка наличия octave

if ! command -v octave &> /dev/null; then

sudo apt install octave

fi

echo "Продолжаем установку ядра..."

if [[ "$USE\_PIPX" == "1" ]]; then

echo "Установка через pipx..."

pipx inject --include-apps --include-deps jupyter octave\_kernel

PIPX\_PATH="$(pipx environment | grep PIPX\_LOCAL\_VENVS | cut -d= -f2)"

source "$PIPX\_PATH"/jupyter/bin/activate

python3 -m octave\_kernel install --user

deactivate

else

echo "Установка через pip..."

pip install --user octave\_kernel

python3 -m octave\_kernel install --user

fi

echo "Ядро Octave успешно добавлено в Jupyter!"

return 0

fi

}

Предпоследний этап — установка ядра Wolfram, процесс установки данного ядра отличается от предыдущих. Для настройки этого ядра потребуется учетная запись в Wolfram Cloud. От пользователя будет необходимо предоставить свой wolfram ID и пароль [57]. Установка осуществляется функцией wolfram\_engine\_install. При запуске установки данного ядра проверяется есть ли в системе пользователя установленный дистрибутив Wolfram Mathematica, либо же Wolfram Engine – бесплатный движок, поддерживающий Wolfram Language. Если алгоритм обнаружил дистрибутив в системе, начнется загрузка конфигурационного скрипта Wolfram Kernel For Jupyter с официального сайта [58], для этого в операционную систему пользователя автоматически будет установлен клиент git, иначе начнется загрузка установочного скрипта Wolfram Engine. При установке Wolfram Engine пользователю необходимо указать директорию, куда будет сохранен дистрибутив. Фрагмент процесса установки Wolfram Engine приведен на рисунке 3.

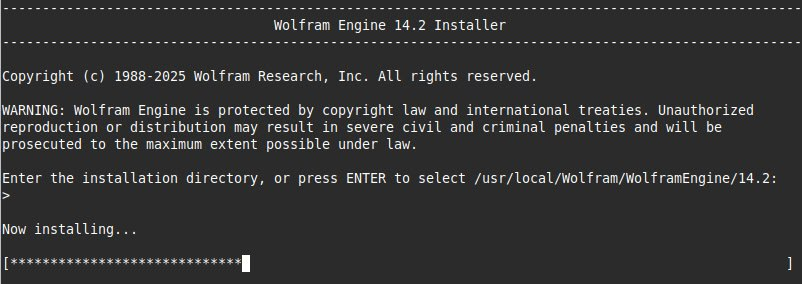


Рисунок 3 – Фрагмент установки wolfram engine

Когда Wolfram Engine будет установлен произойдет загрузка Wolfram Kernel for Jupyter с последующей активацией ядра. В случае успешной активации ядра будет прислано соответствующее сообщение.

Последний этап установки СОМ – это настройка системы модулей для Julia и Python. Для этого используется отдельно написанный скрипт на Julia package\_installer.jl. Данный скрипт вызывается функцией package\_install:

function package\_install() {

USE\_PIPX=$USE\_PIPX julia package\_installer.jl 2>/dev/null || {

echo "Ошибка: Не удалось запустить package\_installer.jl"

return 1

}

}

Здесь проверяется возможно ли запустить скрипт, если нет, то выводится сообщение об ошибке, при этом в *package\_installer* передается информация через что осуществлялась установка python зависимостей.

* 1. Система модулей СОМ

Поддержка модулей одна из ключевых особенностей СОМ. Модули — это библиотеки из экосистем языков Julia и Python. Рассмотрим структуру файла *package\_installer.jl.*

При запуске скрипта вызывается функция *main().* В ней инициализируются массивы *julia\_to\_install* и *python\_to\_install*, в этих массивах будут записаны библиотеки, Julia и Python, которые пользователь выберет для установки. При запуске скрипта пользователю будет выведен список доступных к установке пакетов. Названия пакетов хранятся в виде словарей вида:

название\_категории => массив\_пакетов

в переменных *JULIA\_PACKAGES* и *PYTHON\_PACKAGES* соответственно.

JULIA\_PACKAGES = Dict(

"Стандартный набор" => ["Plots", "IJulia", "Pluto", "PyCall", "SpecialFunctions", "Images"],

"Построение графиков" => ["Makie", "Gadfly", "GLMakie"],

"Разработка приложений с графическим интерфейсом" => ["QML"],

"Дополнительные структуры данных" => ["DataStructures", "StructArrays", "DecisionTree"],

"Дифференциальные уравнения" => ["DifferentialEquations", "ModelingToolkit"],

"Символьные вычисления" => ["Symbolics", "SymbolicUtils", "ModelingToolkit"],

"Методы оптимизации" => ["Optim", "JuMP", "BlackBoxOptim"],

"Интерполяции" => ["Interpolations"],

"Интегрирование" => ["QuadGK", "Roots", "Calculus"],

"Нелинейные уравнения" => ["NLsolve"],

"Стохастический анализ" => [

"CategoricalArrays", "Clustering", "Combinatorics",

"Distributions", "GLM", "StatsBase", "StatsModels", "StatsPlots",

"MultivariateStats", "Measures"

]

)

# Python Packages

PYTHON\_PACKAGES = Dict(

"Стандартный набор" => ["numpy", "pandas", "scipy", "sympy"],

"Визуализация" => ["seaborn", "plotly"],

"Работа с изображениями" => ["opencv-python", "Pillow"],

"Математика высокой точности" => ["mpmath"],

"Машинное обучение и глубокое обучение" => ["scikit-learn", "torch", "tensorflow", "torchvision", "torchaudio"]

)

Модули разделены на группы по категориям При запуске *package\_installer.jl* пользователю будет предложено выбрать среди доступных категорий модули, которые ему необходимы для решения его интересующих задач (рисунок 4).

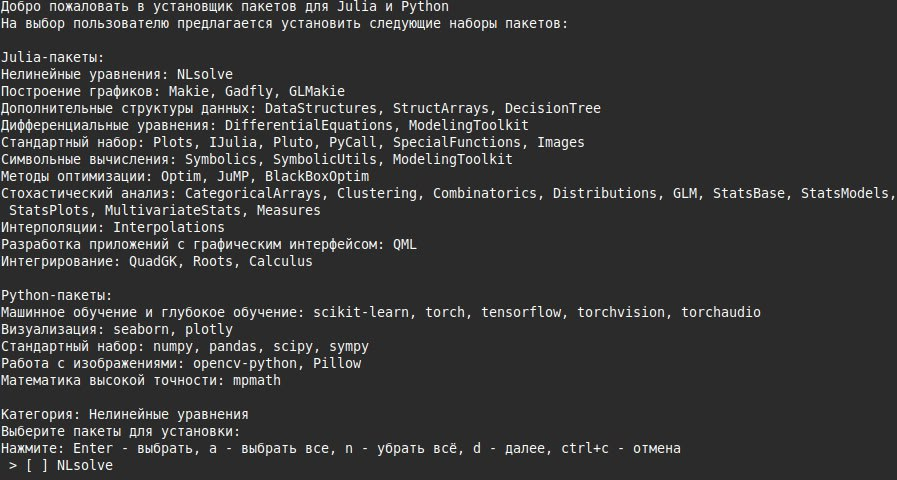


Рисунок 4 – Интерфейс меню настройки модулей

При желании пользователь может изменить данный список, по аналогии вписывая в указанные словари новые категории и соответствующими им пакеты, либо же убрать ненужные. Затем при повторном запуске *package\_installer* автоматически обновятся списки и станет возможна установка новых библиотек. Представленные здесь пакеты являются примером.

После ознакомления со списком доступных пакетов пользователю посредством интерактивного меню будет необходимо выбрать пакеты, которые он желает установить. Реализованное интерактивное меню позволяет выбрать как все пакеты из категории для установки, так и лишь конкретные.

Управление осуществляется на стрелочки и клавиши:

1. a — выбрать все библиотеки из категории;
2. enter — выбрать библиотеку для установки;
3. n — убрать всё из выбранных библиотек категории;
4. d — переключиться на следующую категорию;
5. ctrl+c — выйти из скрипта.

Так как пользователь выбирает, лишь те модули, которые ему нужны, таким образом минимизируется количество лишнего функционала, при этом сохраняется возможность поставить все наборы целиком, если это потребуется.

Кроме того, модифицированную версию *package\_installer.jl* можно распространять среди коллег или например, студентов, тем самым у каждого будет инструмент для создания идентичной среды.

Таким образом реализованный механизм модулей позволяет:

1. легко создать уникальную вычислительную среду с упором на задачи конкретного пользователя;
2. воссоздать уже настроенную среду путем передачи модифицированного скрипта;
3. избавить пользователя от лишних зависимостей.

Полный список пакетов, доступных для установки в СОМ представлен в Приложении А.

* 1. Конфигурация через флаги

Скрипт *installer.sh* поддерживает запуск с флагами, в программе предусмотрены следующие флаги:

1. *--h, --help* — при запуске с этими флагами будет выведено справочное сообщение (рисунок 5).

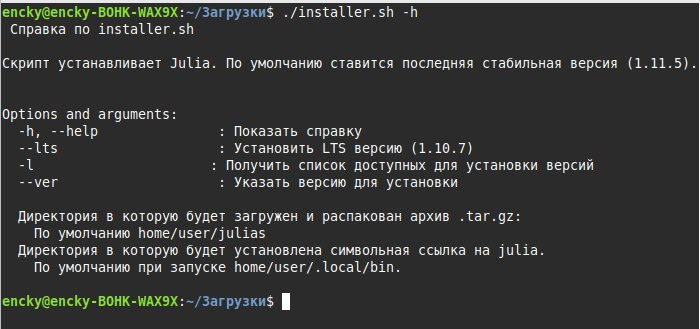


Рисунок 5 – Вывод скрипта с флагом -h

1. *--ver, -v* — флаг, позволяющий указать версию Julia для установки. Например, если запустить скрипт следующим образом *./installer.sh –ver 1.11.2*

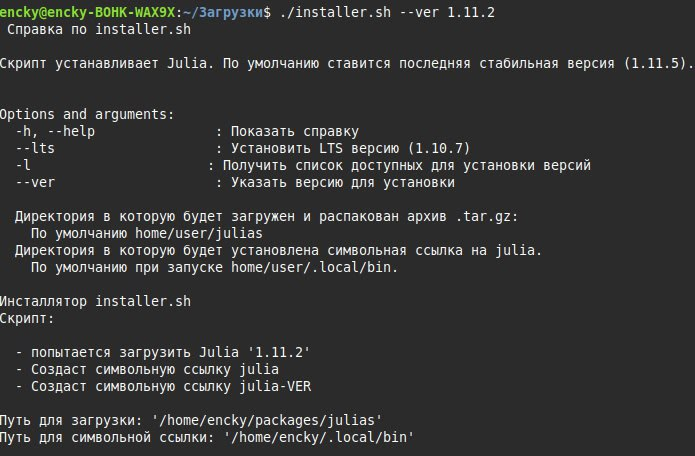


Рисунок 6 – Cмена версии языка

На изображении видно, что после вызова команды скрипт попытается загрузить указанную версию.

1. *-l, --list —* выводит список всех доступных версий Julia для установки, для корректной работы команды требуется пакет *jq*, который будет установлен при первом запуске скрипта.

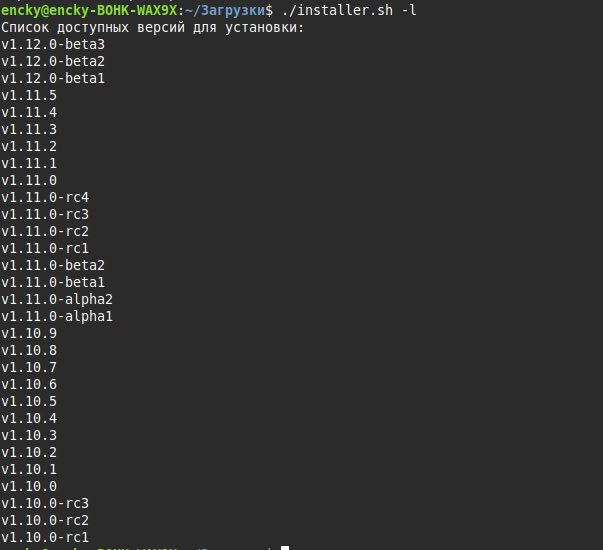


Рисунок 7 – Вывод команды -l

Команды -l и -ver следуют использовать совместно, сначала с помощью -l пользователь выводит список доступных версий, а затем указывает нужную через -ver.

1. Использование СОМ при решении задач

В данной главе будет рассмотрено практическое использование «Системы открытой математики», будут рассматриваться задачи:

1. нахождение всех корней полинома N-ой степени с использованием метода парабол, нахождение всех корней полинома N-ой степени с использованием модифицированного метода касательных;
2. аппроксимация значения числа через нахождение суммы ряда.
   1. Постановка задачи нахождения корней полинома

Уравнение вида , где — многочлен отличный от нуля, называется алгебраическим уравнением и может быть записано относительно в виде:

|  |  |
| --- | --- |
|  | (1) |

где , и — коэффициенты уравнения -ой степени.

Рассмотрим задачу нахождения всех корней данного уравнения используя **метод парабол** и **модифицированный метод касательных**.

* + 1. Алгоритм парабол

Алгоритм состоит из многократного повторения следующих действий:

1. нахождение очередного корня методом парабол [13].
2. понижение порядка полинома путем деления многочлена на [13], [59]−[60].

В методе парабол для вычисления значения полинома в точке используется схема Горнера — рекурсивный способ вычислить значения в точке:

|  |  |  |
| --- | --- | --- |
|  | | (2) |
|  |  | |
|  |  | |
|  | (3) | |
|  |  | |
|  |  | |

Метод парабол позволяет найти все корни полинома, включая действительные и комплексные, и он сходится при любом начальном приближении. Однако для этого требуется проводить вычисления в комплексных числах, а также задать три начальные точки.

В алгоритме задаются точки:

|  |  |
| --- | --- |
|  |  |
|  | (4) |
|  |  |

Через них проводится парабола вида:

|  |  |
| --- | --- |
|  | (5) |

Тогда точка пересечения параболы с осью абсцисс будет следующей точкой приближения корня.

Формула для вычисления точки приближения корня имеет следующий вид:

|  |  |
| --- | --- |
|  | (6) |

При этом:

|  |  |  |
| --- | --- | --- |
|  | | (7) |
|  |  | |
|  | (8) | |
|  |  | |
|  |  | |

Знак в уравнении (6) выбирается таким образом, чтобы знаменатель был максимальным.

Алгоритм продолжается до тех пор, пока не станет меньше точности .

* + 1. Модифицированный метод касательных

Модифицированный метод касательных (модифицированный метод Ньютона) — итерационный метод, решающий задачу нахождения корня уравнения . В отличии от классического метода, который находит только один корень, модифицированный метод находит все корни.

Как и в классическом методе используем итерационную формулу:

|  |  |
| --- | --- |
|  | (9) |

где

— производная функции .

Условие завершения итераций: , где — заданная точность.

В случае если функция задана полиномом она примет следующий вид:

|  |  |
| --- | --- |
|  | (10) |

Здесь — переменная, — коэффициенты полинома, — степень полинома.

Производная полинома:

|  |  |
| --- | --- |
|  | (11) |

Ключевым отличием модифицированного метода от классического является локализация области, где находятся предполагаемые корни.

Для нахождения данной области можно воспользоваться следующими утверждениями, известными из курса математики:

1. все комплексные корни полинома попарно сопряжены;
2. полином нечетной степени имеет, по крайней мере один действительный корень;
3. все корни полинома лежат в круге радиуса где — наибольшее значение из ;
4. все корни полинома лежат за пределами круга радиуса   
    где — наибольшее значение из ;

Так как корни находятся в комплексной плоскости, то все корни находятся внутри кольца с внутренним радиусом и внешним радиусом . Центр кольца — начало координат.

Остается в этой области построить сетку и выбирая каждый узел сетки в качестве начального приближения запускать метод касательных. Полученные решения необходимо проверить на уникальность: если   
, то решение уникально.

* + 1. Аппроксимация значения числа через ряды

Число — известная иррациональная математическая константа, которая используется во многих формулах и расчетах, используемых во многих сферах: в физике, космосе, электротехнике, электронике, теории вероятностей, строительстве, навигации и пр.

В вычислительной математике одним из популярных подходов к вычислению значения числа является аппроксимация числа с использованием последовательностей и степенных рядов, которые сходятся к данному числу или приближенно равны ему.

Для демонстрации возможностей СОМ реализуем следующие методы аппроксимации числа .

*Алгоритм Мэчина.*

Джон Мэчин используя предложил:

|  |  |
| --- | --- |
|  | (12) |

где

раскладывается в ряд

*Формула братьев Чудновских.*

|  |  |
| --- | --- |
|  | (13) |

* 1. Решение задачи нахождения корней полинома

В качестве многочлена для метода парабол будем использовать функцию, которая получается из уравнения путем раскрытия скобок:

|  |  |
| --- | --- |
|  | (14) |

Для метода касательных исследуем корни уравнения:

|  |  |
| --- | --- |
|  | (15) |
|  |  |

Полиномиальный вид которого:

|  |  |
| --- | --- |
|  |  |
| + |  |
|  | (16) |
|  |  |
| = 0. |  |

* + 1. Метод парабол

Рассмотрим реализацию решения данной задачи на:

1. Python;
2. Julia;
3. Scilab при помощи встроенных методов.

Для решения задачи средствами Python нам понадобится библиотеки NumPy и math, cmath. Для реализации на Julia, необходимо воспользоваться библиотекой Combinatorics для генерации коэффициентов полинома.

Описанные функции были реализованы в обоих языках и имеют одинаковые наименования.

Сначала посчитаем коэффициенты полинома, для этого была реализована функция *make\_coeff.*

function make\_coeffs(a::Float64, n::Int)::Vector{ComplexF64}

"""

Генерация коэффициентов полинома (x - a)^n.

"""

coeffs = [binomial(n, k) \* (-a)^(n-k) for k in 0:n]

return reverse(coeffs)

end

Данная функция принимает на вход два параметра: *a* — корень кратности 5, и *n* — степень кратности полинома. Функция возвращает массив комплексных значений.

Далее была реализована функция *polyval* для вычисления значения полинома в точке:

function polyval(P::Vector{ComplexF64}, x::ComplexF64)::ComplexF64

"""

значение полинома P в точке x.

"""

………

end

Данная функция принимает на вход массив коэффициентов полинома, а также точку, в которой необходимо вычислить значение полинома.

Реализованы функции для вычисления конечных разностей первого и второго порядков:

function ras1(P::Vector{ComplexF64}, x0::ComplexF64, x1::ComplexF64)::ComplexF64

return (polyval(P, x1) - polyval(P, x0)) / (x1 - x0)

end

function ras2(P::Vector{ComplexF64}, x0::ComplexF64, x1::ComplexF64, x2::ComplexF64)::ComplexF64

return (ras1(P, x1, x2) - ras1(P, x0, x1)) / (x2 - x0)

end

Здесь входные значения — массив коэффициентов многочлена и — начальные приближения.

В качестве вспомогательной функции реализован алгоритм вычисления одной итерации метода Мюллера:

function muller\_step(P::Vector{ComplexF64}, x0::ComplexF64, x1::ComplexF64, x2::ComplexF64)::ComplexF64

"""

Одна итерация метода Мюллера: три точки x0, x1, x2 -> новое приближение x3

"""

…

end

Которая по имеющимся трем точкам находит следующее приближение. Для выполнения операции деления многочленов используется функция:

function deflate(P::Vector{ComplexF64}, root::ComplexF64)::Vector{ComplexF64}

"""

Деление многочленов

Возвращает новый массив коэффициентов степени на 1 меньше.

"""

……

end

Которая принимает на вход массив коэффициентов полинома и корень уравнения. Результатом выполнения данной функции является новый (уменьшенный) набор коэффициентов.

Функция *find\_root\_muller* отвечает за итерационный поиск корня, поиск осуществляется до тех пор, пока не станет меньше .

И последняя функция — *mullers\_all\_roots* запускает итерационный процесс нахождения всех корней полинома.

Полный код описанных алгоритмов можно найти в репозитории COM [56]. В директории *demo*.

* + 1. Метод касательных

Для решения данной задачи на Python нам потребуются библиотеки math и NumPy. При решении задачи на Julia сторонние библиотеки не требуются.

Зададим необходимую точность итерационного процесса Ньютона, а также количество итераций и погрешность, необходимую для определения близости двух корней уравнения.

NEWTON\_TOL = 1e-8 # точность итераций Ньютона

NEWTON\_MAX\_STEPS = 1000 # макс. кол-во шагов Ньютона

UNIQUE\_TOL = 1e-8 # два корня считаются равными, если ближе

Реализуем функции вычисления — значения полинома в точке и — значения производной функции в точке:

def polynomial\_value(z: complex, coeffs: np.ndarray) -> complex:

return np.polyval(coeffs, z)

def polynomial\_derivative\_value(z: complex, coeffs: np.ndarray) -> complex:

return np.polyval(np.polyder(coeffs), z)

Функции принимают на вход точку и массив коэффициентов полинома . Результатом работы этих функций является комплексные значения и соответственно.

Благодаря функции *deflate\_polynomial* осуществляется процесс деления полиномов методом Горнера.

За поиск всех корней отвечает функция *find\_roots:*

find\_roots(coeffs\_real: list[float]) -> list[complex]

Она принимает на вход коэффициенты полинома и в результате выполнения возвращает массив корней функции (14).

Внутри *find\_roots* проходит основной итерационный процесс, который включает в себя:

1. локализацию области, где расположены корни;
2. построение сетки значений;
3. поиск узлов-решений уравнения.

Полный код программы доступен в git репозитории [56].

* + 1. Аппроксимация числа

Для решения данных задач нам понадобятся библиотека python *—* *mpmath* и встроенная библиотека Julia *—* P*rintf.* Первая необходима для обеспечения быстрых вычислений, вторая для форматированного вывода.

*Формулы Мэчина.*

Алгоритм аппроксимации числа через формулы Мэчина следующий:

1. Задать точность вычислений, в которую входит количество желаемых знаков после запятой + некоторый запас, который необходим для борьбы с потерями округления. В качестве запаса можно взять 40 цифр.
2. Задать константы алгоритма.
3. Посчитать необходимые суммы.

*Формула Чудновских.*

Для аппроксимации числа формулой Чудновских так же как и для формулы Мэчина необходимо:

1. задать точность вычислений и запас;
2. задать константы;
3. пересчитать нужные суммы.

Алгоритм можно ускорить, заменив вычисления факториалов на рекуррентные соотношения.

* 1. Результаты тестов СОМ

Рассмотрим результаты решения задач, реализованные на разных языках. В рамках демонстрации возможностей СОМ в работе рассматривались две задачи:

1. нахождение корней полинома высокой степени;
2. аппроксимация значений числа .
   * 1. Метод парабол

Реализованный метод парабол успешно решает задачу нахождения всех корней полинома для уравнения (14). Ниже приведены корни, полученные в Julia, Python и Scilab.

Результаты полученные в Julia:

5-element Vector{ComplexF64}:

1.1002997445611942 + 0.008889576064149394im

1.0916384745571326 + 0.003032326416413989im

1.1085473748873975 + 0.002462125503724224im

1.1049828005257747 - 0.007368724518387231im

1.0945316054685006 - 0.007015303465900376im

Результаты полученные в Python:

Найденные корни:

1.0986018710448326 + -0.0002006393671534958j

1.0993771798790124 + 0.0012677792106174555j

1.1012489071201006 + -0.0006595424020149749j

1.1010132792358196 + 0.0009840111159257771j

1.0997587627202339 + -0.001391608557374762j

Корни, посчитанные в Scilab встроенными инструментами:

ans =

1.1010545 + 0.0007666i

1.1010545 - 0.0007666i

1.0995968 + 0.001239i

1.0995968 - 0.001239i

1.0986975 + 0.i

Как видно из результатов корни получились близкие друг к другу, значит задача решена верно.

* + 1. Метод касательных

Корни полинома (16), найденные через реализацию на Python:

Найдено корней: 11

root #1: (0.9099999353886055+0j)

root #2: (0.9300003583438123+0j)

root #3: (0.9499994112987741+0j)

root #4: (1.0000019343534863+0j)

root #5: (1.0199956271341701+0j)

root #6: (1.030002851708563+0j)

root #7: (1.069999881772548+0j)

root #8: (200.00001236328276+0j)

root #9: (200.29995593822667+0j)

root #10: (200.5000369633223+0j)

root #11: (200.99999473516837+0j)

Решения, найденные через алгоритм на Julia:

Найдено корней: 11

root #1: 0.9099999484803867 + 0.0i

root #2: 0.9300003036177352 + 0.0i

root #3: 0.9499994807910818 + 0.0i

root #4: 1.0000018059957385 + 0.0i

root #5: 1.0199958613623579 + 0.0i

root #6: 1.030002714204663 + 0.0i

root #7: 1.0699998855479969 + 0.0i

root #8: 200.00002052655634 + 0.0i

root #9: 200.29992668488978 + 0.0i

root #10: 200.50006159777647 + 0.0i

root #11: 200.99999119077748 + 0.0i

Корни, найденные через Scilab:

200.99999 + 0.i 200.50004 + 0.i 200.29995 + 0.i 200.00001 + 0.i 1.0699999 + 0.i 1.0300015 + 0.i 1.0199978 + 0.i 1.0000009 + 0.i 0.9499998 + 0.i 0.9300001 + 0.i 0.91 + 0.i

Полученные результаты говорят о том, что задача успешно решена во всех использованных математических ядрах, при этом результаты между собой очень близки.

* + 1. Аппроксимация числа

В работе рассматривались два метода приближенного вычисления значения числа . Было проведено несколько измерений скорости выполнения программ, с целью определения насколько использованный метод эффективен для аппроксимации числа . Было подсчитано сколько времени алгоритмам требуется для аппроксимации с точностью до ­­­.где . Для каждого проводилось 10 измерений что бы получить среднее время для аппроксимации с заданной точностью. Оценим полученные результаты.

*Формулы Мэчина.*

Результаты вычислений по формуле (12), полученные с помощью языка Python:

digits | avg time, s | iters

----------------------------------

50 | 0.000043 | 35

100 | 0.000049 | 71

200 | 0.000091 | 142

500 | 0.000501 | 356

1000 | 0.002020 | 714

2000 | 0.006033 | 1429

где

*digits* *—* количество верных знаков числа ;

*avg time —* среднее время в секундах, затраченное на достижение указанной точности;

*iters* —минимальное количество итераций, которое потребовалось для достижения заданной точности;

Ниже представлены результаты аппроксимации формулами Мэчина (12) с использованием Julia:

Цифры | среднее время, с

--------------------------

50 | 0.000015

100 | 0.000016

200 | 0.000025

500 | 0.000045

1000 | 0.000088

2000 | 0.000132

Здесь *цифры —* количество верно аппроксимированных цифр числа , *среднее время, с* *—* среднее время в секундах, которое потребовалось программе что бы получить необходимую точность.

Из значений видно, что реализация на Julia работает быстрее алгоритма на Python и чем выше требуемая точность, тем быстрее Julia.

*Формула Чудновских.*

В результате работы алгоритма на Julia были получены следующие результаты:

цифры | итерации | среднее время, s

--------------------------------

10 | 1 | 0.000015

50 | 4 | 0.000024

100 | 8 | 0.000046

200 | 15 | 0.000096

500 | 36 | 0.000328

1000 | 71 | 0.000915

2000 | 142 | 0.010019

Из таблицы видно, что алгоритм достигает высокой точности приближения за небольшое количество итераций. Данный позволяет вычислять значения числа с высокой точностью.

При реализации алгоритма на Python, получились следующие результаты:

цифры | итерации | среднее время

50 знаков | 4 итерац.| 0.0002 сек

100 знаков | 8 итерац.| 0.0005 сек

200 знаков | 15 итерац.| 0.0005 сек

500 знаков | 36 итерац.| 0.0019 сек

1000 знаков | 71 итерац.| 0.0089 сек

2000 знаков | 142 итерац.| 0.0398 сек

Из полученных данных видно, что Python и здесь отстает от Julia, что делает Julia более предпочтительной для решения сложных вычислительных задач.

В ходе работы были рассмотрены два класса вычислительных задач, для решения которых возможно использование СОМ в качестве основного инструмента. Были рассмотрены возможности некоторых модулей, которые входят в комплект модулей по умолчанию. Таким образом, была показана работоспособность проекта и его возможности при решении сложных вычислительных задач.

Заключение

В результате работы была разработана технология для создания свободных математических пакетов с заданными свойствами, обладающий гибкими возможностями настройки и широкой областью применения. На базе разработанной технологии был построен математический пакет с поддержкой вычислительных ядер Julia, Python, Scilab, Octave, Wolfram Language и Bash.

Реализована система модулей, позволяющая устанавливать наборы специализированных библиотек для языков Julia и Python.

Данный продукт позволит ученым, исследователям, студентам решать различные задачи в рамках одной среды, а также предоставляет возможность создать уникальную математическую среду с заданными пользователем свойствами.

Описанный метод создания свободных математических пакетов может быть использован для создания других проектов, что позволяет использовать данный проект для новых математических пакетов.

Работа была представлена на двух научных конференциях в Смоленске («Системы компьютерной математики и их приложения») в 2024 году и в Переславле-Залесском («Свободное программное обеспечение у высшей школе») в 2025 году, где вызвала дискуссию и положительный отклик среди профессионального сообщества, что подтверждает актуальность выбранной темы. По мотивам выступлений были опубликованы научные статьи [59]−[60], вошедшие в сборники трудов конференций, а также издана статья [61] в сборнике работ («Алгебра и приложения») факультета Математики и компьютерных наук.

Проект был протестирован на операционных системах: Kubuntu 24.04.

Список использованных источников

1. Wolfram Mathematica: Современные технические вычисления: сайт. – URL: https://www.wolfram.com/mathematica/ (дата обращения: 02.05.2025).
2. Mathematical Typesetting–Wolfram Language Documentation: сайт. – URL:https://reference.wolfram.com/language/guide/MathematicalTypesetting.html (дата обращения: 02.05.2025).
3. Егорова Д. К. О реализации параллельных вычислений в системе Wolfram Mathematica / Д. К. Егорова, М. Д. Шмелева // Огарёв-Online. 2016. №20 (85): сайт. – URL: https://cyberleninka.ru/article/n/o-realizatsii-parallelnyh-vychisleniy-v-sisteme-wolfram-mathematica (дата обращения: 02.05.2025).
4. The Wolfram Solution for Biosciences: сайт. – URL: https://www.wolfram.com/biosciences/ (дата обращения: 03.05.2025).
5. Пономарева, С. В. О применении прикладных математических пакетов для решения экономических задач / С. В. Пономарева, О. Н. Пыжкова, И. М. Борковская // Информатизация образования и методика электронного обучения: цифровые технологии в образовании: материалы IV Международной научной конференции. – Красноярск: Сибирский федеральный университет, 2020. – С. 323-327.
6. Scilab | Scilab: сайт. – URL: https://www.scilab.org/ (дата обращения: 04.05.2025).
7. Дохтаева, И. А. Определение достоинств среды программирования SciLab путем сравнительного анализа языков программирования MatLab и SciLab / И. А. Дохтаева, Е. Н. Рожина // Прогрессивные технологии и процессы: Сборник научных статей Международной молодежной научно-практической конференции. – Курск: Закрытое акционерное общество «Университетская книга», 2014. – С. 204-207.
8. Стрелков, Н. О. Моделирование радиотехнических сигналов и цепей в Scilab и Xcos / Н. О. Стрелков // Инфорино-2018: материалы IV международной научно-практической конференции. – Москва: Национальный исследовательский университет «МЭИ», 2018. – С. 338–343.
9. Исследование эффективности средств пакета Scilab при решении задач оптимизации / Т. И. Семенова, В. Н. Шакин, В. В. Фриск, А. Ю. Кудряшова // СВЧ-техника и телекоммуникационные технологии. – 2020. – № 1–2. – С. 136–137.
10. ATOMS: Homepage: сайт. – URL: <https://atoms.scilab.org/> (дата обращения: 04.05.2025).
11. Анисимова, Э. С. Особенности применения программной среды Scilab в преподавании дисциплины «численные методы» // Экономика и социум. 2016. №12-3 (31): сайт. – URL: <https://cyberleninka.ru/article/n/osobennosti-primeneniya-programmnoy-sistemy-scilab-v-prepodavanii-distsipliny-chislennye-metody> (дата обращения: 28.05.2025).
12. Андреев, С. М. Математическое моделирование систем управления и их элементов: Электронный ресурс / С. М. Андреев, Д. В. Чистяков. – Магнитогорск: Магнитогорский государственный технический университет им. Г.И. Носова, 2021. – URL: https://www.elibrary.ru/item.asp?id=47950103 (дата обращения: 28.05.2025).
13. Алексеев, Е. Р. Scilab: Решение инженерных и математических задач: учеб. издание. / Е.Р. Алексеев, К. В. Дога, О. В. Чеснокова М.: Базальт СПО; ДМК Пресс, 2024. – 440 с.
14. SageMath – Open-Source Mathematical Software System: сайт. – URL: <https://www.sagemath.org/> (дата обращения: 07.05.2025).
15. Packages and Features: сайт. – URL: <https://doc.sagemath.org/html/en/reference/spkg/#packages-and-features> (дата обращения: 08.05.2025).
16. Cython: C-Extensions for Python: сайт. – URL: <https://cython.org/> (дата обращения: 08.05.2025).
17. Ашуркин, Д. П. Система компьютерной алгебры SAGE как средство для решения систем уравнений / Д. П. Ашуркин, Н. Ю. Богданова // Молодой исследователь Дона. 2016. №2 (2). – С. 115-120.
18. Голубков, А. Ю. Компьютерная алгебра в системе SAGE / А. Ю. Голубков, А. И. Зобнин, О. В. Соколова. – Москва : Московский государственный технический университет им. Н.Э. Баумана, 2013. – 80 с.
19. MathWorks — Maker of MATLAB and Simulink - MATLAB & Simulink: сайт. – URL: https://www.mathworks.com/ (дата обращения: 09.05.2025).
20. Products and Services — MATLAB & Simulink: сайт. – URL: https://www.mathworks.com/products.html (дата обращения: 10.05.2025).
21. Викович, И. А. Моделирование динамических процессов в колесных машинах средствами Matlab Simulink и Matlab Simulink/Simscape / И. А. Викович, О. В. Житенко, М. М. Осташук // Вiсник СевНТУ. – 2012. – № 134. – С. 200-204. – URL: https://www.elibrary.ru/item.asp?id=23525690 (дата обращения: 10.05.2025).
22. Берков, Н. А. Математический практикум с применением пакета MatLab: Учебное пособие / Н. А. Берков, А. И. Архангельский, М. В. Архангельская. Том Часть 1. – Москва: Учреждение высшего образования «Институт системных технологий», 2017. – 74 с. – URL: https://www.elibrary.ru/item.asp?id=28344749 (дата обращения: 10.05.2025).
23. Mathcad: Math Software for Engineering Calculations | Mathcad: сайт. – URL: https://www.mathcad.com/ (дата обращения: 06.05.2025).
24. Пшеничнов, Ю. А. Среда Mathcad как средство изучения математики при подготовке инженеров / Ю. А. Пшеничнов, Е. А. Задорожнюк // Научные и методические аспекты математической подготовки в университетах технического профиля: Материалы V Международной научно-практической конференции, посвященной 70-летию БелГУТа, Гомель, 27 апреля 2023 года. – Гомель: Учреждение образования "Белорусский государственный университет транспорта", 2023. – С. 167-170.
25. Краморов, А. В. Практическое применение математического пакета Mathcad при выполнении инженерных расчетов / А. В. Краморов // Информационный бюллетень Омского научно-образовательного центра ОмГТУ и ИМ СО РАН в области математики и информатики : материалы IX Международной молодежной научно-практической конференции с элементами научной школы «Прикладная математика и фундаментальная информатика», посвященной 80-летию со дня рождения академика РАН Евтушенко Ю. Г. – Омск: Омский государственный технический университет, 2019. – С. 189-191.
26. Рындин, В. В. Применение системы Mathcad при статистическом анализе экспериментальных данных / В. В. Рындин, Л. Ю. Волкова // Наука и техника Казахстана. – 2018. – № 4. – С. 6-18.
27. Mathcad Resources | Mathcad: сайт. – URL: https://www.mathcad.com/en/resource-center (дата обращения: 12.05.2025).
28. R: The R Project for Statistical Computing: сайт. – URL: https://www.r-project.org/ (дата обращения: 13.05.2025).
29. Величко, В. В. Сравнительный анализ статистических пакетов программ // Инновационная наука. 2016. №5-2 (17): сайт. – URL: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-statisticheskih-paketov-programm (дата обращения: 13.05.2025).
30. Андропова, О. Ю. Использование R-пакета для исследования устойчивости сложных систем / О. Ю. Андропова // Системы управления, сложные системы: моделирование, устойчивость, стабилизация, интеллектуальные технологии : материалы VII Международной научно-практической конференции. – Елец: Елецкий государственный университет им. И.А. Бунина, 2021. – С. 50–54.
31. Геник, И. В. Возможности использования языка программирования R в задачах геологии и геофизики / И. В. Геник // Горное эхо. – 2022. – № 4(89). – С. 68–74.
32. Крисяк, М. И. Использование языка R в анализе данных и его преимущество при создании геоинформационной подсистемы / М. И. Крисяк, А. Г. Бурда // Научное обеспечение агропромышленного комплекса : Сборник статей по материалам 79-й научно-практической конференции студентов по итогам НИР за 2023 год. – Краснодар: Кубанский государственный аграрный университет им. И.Т. Трубилина, 2024. –   
    С. 268–269.
33. Cran: Contributed Packages: сайт. – URL: https://cran.r-project.org/web/packages/ (дата обращения: 02.05.2025).
34. Overview: сайт. – URL: https://journal.r-project.org/index.html (дата обращения: 15.05.2025).
35. Welcome to Python.org: сайт. – URL: https://www.python.org/ (дата обращения: 17.05.2025).
36. Отаханов, Н. А. Преподавание языка программирования Python в высших учебных заведениях / Н. А. Отаханов // Информатика и образование. – 2023. – Т. 38, № 5. – С. 78-86. – URL: https://elibrary.ru/item.asp?id=54830316 (дата обращения: 17.05.2025).
37. Панова, И. В. Методические аспекты обучения программированию на языке Python в школьном курсе информатики / И. В. Панова, А. А. Коливнык // Информатика в школе. – 2020. – № 6(159). – С. 47-50. – URL: https://elibrary.ru/item.asp?id=44035430 (дата обращения: 02.05.2025).
38. Матисаков, Ж. К. Моделирование определения длины световой волны с помощью дифракционной решетки на Python / Ж. К. Матисаков, Ж. И. Мамбетов, Ж. К. Матисаков // Бюллетень науки и практики. – 2024. – Т. 10, № 7. – С. 21–25.
39. PyPI · The Python Package Index. – URL: https://pypi.org/ (дата обращения: 17.05.2025).
40. Введение в машинное обучение и его применение на Python / И. И. Кушалиева, М. М. Ахмарова, Л. И. Кушалиева, О. М. Минаев // Актуальные вопросы физико-математического образования : Материалы межрегиональной студенческой научно-практической конференции, Грозный, 29 апреля 2021 года. – Грозный: Чеченский государственный педагогический университет, ИП Овчинников Михаил Артурович (Типография Алеф), 2021. – С. 460-464. – URL: https://elibrary.ru/item.asp?id=46347016 (дата обращения: 18.05.2025).
41. The Julia Programming Language: сайт. – URL: https://julialang.org/ (дата обращения: 19.05.2025).
42. JIT-компиляция – Википедия: сайт. – URL: https://ru.wikipedia.org/wiki/JIT-компиляция (дата обращения: 02.05.2025).
43. Зырянов, Д. М. Язык научных вычислений Julia / Д. М. Зырянов, А. А. Емельянов, К. С. Лещенко // Оригинальные исследования. – 2020. – Т. 10, № 9. – С. 50-56. – URL: https://elibrary.ru/item.asp?id=44069307 (дата обращения: 18.05.2025).
44. stepanzh – Сердце языка Julia – множественная диспетчеризация: сайт. – URL: https://stepanzh.github.io/julia-multiple-dispatch.html (дата обращения: 19.05.2025).
45. Julia Packages: сайт. – URL: https://www.juliapackages.com/ (дата обращения: 18.05.2025).
46. Oceananigans — Julia Packages: сайт. – URL: https://www.juliapackages.com/p/oceananigans (дата обращения: 19.05.2025).
47. GitHub - SciML/ModelingToolkit.jl: An acausal modeling framework for automatically parallelized scientific machine learning (SciML) in Julia. A computer algebra system for integrated symbolics for physics-informed machine learning and automated transformations of differential equations: сайт. – URL: https://github.com/SciML/ModelingToolkit.jl (дата обращения: 15.05.2025).
48. GitHub — JuliaSymbolics/Symbolics.jl: Symbolic programming for the next generation of numerical software: сайт. – URL: https://github.com/JuliaSymbolics/Symbolics.jl (дата обращения: 12.05.2025).
49. Оконешникова, Е. А. Научное программирование на языке julia: распараллеливание фрагмента кода для улучшения производительности / Е. А. Оконешникова // Новые информационные технологии в образовании и науке. – 2021. – № 4. – С. 81-86.
50. Рожков, А. В. Экспериментальная математика на платформе языка программирования Julia / А. В. Рожков // Материалы Всероссийской научно-практической конференции «тенденции и перспективы развития обучения математике и информатике в условиях реальной и цифровой среды», Краснодар, 28 марта 2023 года. – Краснодар: Кубанский государственный университет, 2023. – С. 30-36.
51. Рожков, А. В. Обучение и научные исследования в области математики и информатики на платформе языка Julia / А. В. Рожков, Е. А. Оконешникова // Новые информационные технологии в образовании и науке. – 2020. – № 3. – С. 87-91.
52. Моисеева, Н. А. Возможности Jupyter Notebook для конструирования цифровых интерактивных учебных материалов / Н. А. Моисеева // Цифровые инструменты в образовании : Сборник статей по материалам Всероссийской научно- практической конференции с международным участием. – Сургут: Сургутский государственный педагогический университет, 2023. – С. 62-64.
53. Головатин, Д. Т. Разработка свободной среды обработки математических данных на основе языка программирования Julia. Выпускная квалификационная работа (Бакалаврская работа) / Головатин Давид Тимурович; Кубанский Государственный Университет. ‒ Краснодар, КубГУ, 2023.
54. Ке­­тов, Д. В. Внутреннее устройство Linux: / Д. В. Кетов. ‒ 2-е изд., перераб. и доп. ‒ СПб. : БХВ-Петербург, 2021. ‒ 400 с. ‒ ISBN 78-5-9775-6630-8.
55. Шоттс, У. Командная строка Linux. Полное руководство. / У. Шоттс. ‒ СПб. : Питер, 2017 ‒ 480 с. ‒ ISBN 978-5-496-02303-0.
56. GitHub – EnckyOff/SysOpenMath: сайт. – URL: https://github.com/EnckyOff/SysOpenMath (дата обращения: 21.05.2025).
57. Wolfram Cloud: сайт. – URL: https://www.wolframcloud.com/ (дата обращения: 19.05.2025).
58. GitHub – WolframResearch/WolframLanguageForJupyter: сайт. – URL: https://github.com/WolframResearch/WolframLanguageForJupyter (дата обращения: 16.05.2025).
59. Jupyter как универсальная среда для обучения и научных исследований / Е. Р. Алексеев, Е. А. Вербичева, К. В. Дога [и др.] // Системы компьютерной математики и их приложения. – 2024. – № 25. – С. 3-9. – URL: https://www.elibrary.ru/item.asp?id=74211481 (дата обращения: 19.05.2025).
60. Инструменты разработки специализированных математических пакетов / Д. Т. Головатин, Е. Р. Алексеев, Е. А. Вербичева, К. В. Дога // Системы компьютерной математики и их приложения. – 2024. – № 25. – С. 15-20.
61. Головатин, Д. Т. Интерактивные блокноты семейства Jupyter: универсальная среда для обучения и разработки / Д.Т. Головатин, И.А. Горбунова, П. П. Чепурко, Ю. Р. Чуракова, Е. Р. Алексеев, Ю. С. Коваленко // Алгебра и приложения: сборник научных трудов. – Краснодар Кубанский государственный университет. – Краснодар: Кубанский государственный университет, 2025. – С. 147–161.

ПРИЛОЖЕНИЕ А

**Краткое описание устанавливаемых пакетов**

Модули Julia:

1. Plots — модуль для рисования графиков;
2. IJulia — ядро для Jupyter;
3. Pluto — браузерный блокнот от сообщества Julia;
4. PyCall — модуль для вызова кода на python, позволяет использовать библиотеки, написанные на Python;
5. SpecialFunctions — модуль, добавляющий известные математические функции;
6. Images — модуль для работы с изображениями;
7. QuadGK — модуль с функционалом одномерного численного интегрирования;
8. Roots — модуль с функционалом поиска корней;
9. Calculus — модуль, добавляющий базовый инструментарий для решения задач интегрального и дифференциального исчисления;
10. Symbolics — модуль, реализующий поддержку символьной арифметики;
11. SymbolicsUtils — вспомогательный модуль для Symbolics;
12. ModelingToolkit — фреймворк, расширяющий возможности Julia до полноценной системы символьно-числового моделирования;
13. Optim — модуль, добавляющий функционал для одномерной и многомерной оптимизации;
14. JuMP — предметно-ориентированный язык для математической оптимизации;
15. BlackBoxOptim — модуль, добавляющий методы безградиентной оптимизации;
16. Interpolations — модуль, добавляющий интерполяционные методы;
17. NLSolve — модуль для решения нелинейных уравнений;
18. Makie — модуль для визуализации данных, альтернатива Plots;
19. Gradfly — еще один модуль для визуализации, альтернатива Plots;
20. GLMakie — вспомогательный модуль для Makie;
21. QML — Обертка для QT;
22. DifferentialEquations — модуль для решения дифференциальных уравнений;
23. CategoricalArrays — модуль, добавляющая категориальные массивы;
24. Clustering — модуль для кластерного анализа;
25. Combinatorics — модуль, для решения комбинаторных задач;
26. Distribution — модуль, для работы с распределениями;
27. GLM — стохастическая библиотека для работы с линейными и обобщенными моделями;
28. StatsBase — модуль для работы с статистическими величинами;
29. StatsModels — модуль включает в себя сновные функции для определения, подбора и оценки статистических моделей в Julia;
30. StatsPlots — модуль, упрощающий построение статистических графиков;
31. MultivariableStats — модуль для решения задач многомерной статистики, например, уменьшения размерности;
32. Measures — модуль, добавляющий различные унифицированные единицы измерения;
33. DataStructures — модуль, добавляющий дополнительные структуры данных;
34. StructArrays — добавляет массивы структур;
35. DecisionTree — добавляет решающие деревья.

Модули Python:

1. NumPy — модуль для научных вычислений, предоставляет поддержку многомерных массивов и матриц, а также математические функции для операций над ними;
2. Pandas — модуль для работы с таблицами и временными рядами
3. SciPy — модуль, предоставляющий широкие возможности для решения вычислительных задач оптимизации, интерполяции, численного дифференцирования и др.;
4. SymPy — модуль для решения задач символьной алгебры;
5. Seaborn — модуль для визуализации статистических данных;
6. Plotly — модуль, позволяющий создавать интерактивную визуализацию;
7. Opencv-python — python обертка над OpenCV. Библиотека для решения задач машинного зрения;
8. Pillow — библиотека для манипуляции изображениями;
9. Mpmath — модуль, добавляющий возможность работать с арифметикой высокой точности;
10. Scikit-learn — библиотека для решения задач машинного обучения и статистики;
11. Torch — фреймворк для решения задач глубокого обучения;
12. TensorFlow — фреймворк для решения задач глубокого обучения;
13. Torchvision — дополнительный модуль к Torch, предоставляющий наборы данных для компьютерного зрения, модели для классификации, сегментации, детекции объектов, а также функции для преобразования изображений;
14. Torchaudio — модуль к Torch предоставляющий наборы данных и функции для обработки аудиосигналов.