УДК 004.942:519.682 DOI: 10.18698/0236-3933-2020-2-27-43

# О ВОЗМОЖНОСТЯХ ИСПОЛЬЗОВАНИЯ ЯЗЫКА ПРОГРАММИРОВАНИЯ JULIA ДЛЯ РЕШЕНИЯ НАУЧНЫХ И ТЕХНИЧЕСКИХ ЗАДАЧ

**Г.В. Белов**[1] gbelov@yandex.ru
**Н.М. Аристова**[2] aristo2012@yandex.ru

[1] **МГУ им. М.В. Ломоносова, Москва, Российская Федерация**
[2] **ОИВТ РАН, Москва, Российская Федерация**

**Аннотация**

Приведено краткое описание языка Julia, рассмотрены некоторые прикладные библиотеки и программные платформы, написанные на этом языке, предназначенные для решения задач из разных областей науки и техники. Язык Julia относительно молод, однако проведенный анализ литературных источников свидетельствует о его растущей популярности. Представлен обзор лишь небольшого числа библиотек и программных платформ, которые могут быть полезны научным сотрудникам и инженерам, связанным с разработкой средств моделирования и проведением расчетов на компьютере. В частности, не рассмотрены специализированные библиотеки, предназначенные для исследований в области экономики, биологии и медицины. В настоящее время существует большое число языков программирования, причем каждый создавался для того, чтобы преодолеть недостатки уже имеющихся языков. Каждый язык применяется в своей области и имеет свои плюсы и минусы. Цель настоящей работы — ознакомить читателей с возможностями языка программирования Julia для решения научных и технических задач

**Ключевые слова**

*Язык программирования Julia, математическое моделирование, прикладная библиотека*

Поступила 06.02.2020
Принята 10.02.2020
© Автор(ы), 2020

**Введение.** Нужно ли умение программировать инженерам и научным сотрудникам? Ответ на это вопрос не так прост. С одной стороны, сегодня есть множество специализированных программ, предназначенных для проведения расчетов практически во всех областях науки и техники. С другой стороны, как справедливо отмечается в работе [1], всегда есть простые задачи, решать которые с использованием имеющихся программ

неудобно. Кроме того, зачастую возникает необходимость проверить новую идею. Наконец, возникает вопрос: кто будет разрабатывать следующее поколение научных и инженерных программ для компьютера?

Как отмечается в статье [2], долгое время основными языками высокопроизводительных вычислений были Fortran, C и C++. Причинами этого являются их совместимость, способность компиляции в высокоэффективный машинный код и достаточный уровень абстракции по сравнению с программированием на языке ассемблера. Эти языки компилируются в автономном режиме и имеют строгую типизацию переменных, что позволяет обеспечить выполняемую компилятором оптимизацию кода. Позже возник другой класс более современных языков программирования высокого уровня, которые в настоящее время также очень популярны для выполнения научных вычислений. Эти языки, как правило, являются интерпретируемыми, они позволяют разрабатывать приложения с минимальными трудозатратами, однако при этом в большинстве случаев быстродействие программ, написанных на этих языках, невелико. Поэтому интерпретируемые языки зачастую используются для создания кода, который связывает воедино высокопроизводительные модули, написанные на языках типа C и Fortran. В качестве примера можно привести динамические языки программирования, такие как Pyton и R (динамические языки позволяют писать программы, не указывая явно типы переменных, в отличие от языков со статической типизацией). Главными достоинствами подобных языков являются их относительная простота, открытость, кросс-платформенность, возможность быстрой разработки приложений и наличие большого числа бесплатных специализированных библиотек с открытыми текстами, которые образуют так называемую экосистему языка и легко устанавливаются на компьютер пользователя.

Распространение интерпретируемых языков привело к возникновению двухязыковой модели разработки приложений: с использованием языка высокого уровня зачастую отрабатывается алгоритм, который затем реализуется на другом языке, например, Fortran, C или C++. Следствием такого подхода является то, что алгоритм разрабатывают одни люди, а реализуют другие. До настоящего времени именно C и Fortran используются для разработки приложений, требующих трудоемких расчетов с высокой скоростью вычислений. Однако разработка приложений на этих языках является трудоемкой и требует высокой квалификации программиста.

Сравнительно недавно был предложен новый язык программирования Julia (julialang.org) [3, 4], разработчики которого попытались соединить в одном языке высокие быстродействие вычислений и эффектив-

ность разработки приложения. Язык Julia — это кросс-платформенный компилируемый свободно распространяемый язык программирования (лицензия MIT) с динамической типизацией, который имеет ряд достоинств и недостатков. В Julia реализована возможность JIT-компиляции на основе LLVM [5]. Компиляция Just-in-Time (JIT) позволяет обеспечить одновременно выразительность современных интерпретируемых языков и производительность таких языков, как C и Fortran. Компилятор JIT выполняет компиляцию во время первого запуска программы, извлекая из текста информацию, не указанную явно программистом, и используя эту информацию для оптимизации создаваемого машинного кода.

К достоинствам языка Julia можно отнести следующее. Простота — интуитивно понятный язык, синтаксис которого напоминает синтаксис Python и MATLAB; быстродействие вычислений программ, написанных на Julia, сопоставимо с быстродействием программ, написанных на C или Fortran; эффективная векторизация и распараллеливание вычислений, большое число типов данных, включая рациональный и комплексные числа; проведение расчетов при отсутствии некоторых данных (есть тип данных missing), задание точности вычислений (тип данных BigFloat, функция setprecision), реализация символьных вычислений с помощью макрокоманд; использование библиотек, написанных на C и Fortran, и обмен библиотеками с Python и R; интеграция с СУБД (PostgreSQL, MySQL, JSON), поддержка символов Unicode; использование парадигмы множественной диспетчеризации — вызов функции слабо зависит от типа параметров функции (параметрический полиморфизм), наличие хорошей встроенной математической библиотеки с функциями линейной алгебры.

Julia — это полноценный язык программирования, в котором есть целочисленная арифметика, операции с плавающей точкой, циклы, работа с C-подобными структурами и даже аналог оператора goto. Вместе с тем в языке существует очень удобная возможность векторизации расчетов, которая характерна для языков высокого уровня.

Если говорить о недостатках языка Julia, то следует отметить, что время компиляции больших программ может быть довольно заметным (минуты); язык относительно новый, поэтому число библиотек на этом языке относительно невелико (по сравнению с Python); невозможно создать отдельно исполняемую программу — программы на языке Julia работают только в среде Julia.

**Библиотеки и программные платформы.** Язык Julia позиционируется разработчиками как язык для решения научных и технических задач. Его широко используют математики. В работе [6] приведена написанная

на языке Julia библиотека Optim.jl, предназначенная для решения задач оптимизации. Библиотека поддерживает вычисления с использованием комплексных чисел и расчеты с заданной точностью. В ней реализованы методы оптимизации, не использующие производные функции, а также методы оптимизации первого и второго порядков. Исследователь имеет возможность задавать производные в программе или они могут быть вычислены численно методом конечных разностей. К моменту написания работы [6] в библиотеке были реализованы главным образом методы оптимизации без ограничений, которые использовались для решения научных и инженерных задач.

В работе [7] речь идет об открытом предметно-ориентированном встроенном в Julia языке моделирования JuMP, с помощью которого исследователи могут решать задачи оптимизации (линейные, нелинейные, с ограничениями и без). Как показано в [7], JuMP позволяет решать задачи оптимизации очень эффективно, благодаря особенностям языка Julia.

В конце 1970-х гг. были созданы первые языки алгебраического моделирования (AML), предназначенные для описания задач оптимизации в удобной для исследователя форме. При этом сам язык алгебраического моделирования не решает задачу оптимизации, а обращается для ее решения к специальной программе (решателю), т. е. основная задача AML передать исходные данные решателю в понятной для него форме, при этом исследователь задает эти данные в виде, удобном для него. Одним из наиболее известных алгебраических языков моделирования является AMPL, который позволяет создавать быстродействующие программы, но не всегда удобен в использовании.

Язык JuMP также является языком алгебраического моделирования, который встроен в Julia. Разработчики постарались предоставить исследователю возможность взаимодействия с решателем, что дает возможность контролировать процесс вычислений и позволяет сократить время подготовки данных при изменении параметров модели.

Первая версия JuMP (2013) поддерживала линейную и смешанную целочисленную оптимизацию. В следующих версиях была обеспечена поддержка квадратичных, конически квадратичных, полуопределенных и общих задач нелинейной оптимизации с использованием производных.

Алгебраический язык моделирования является важным компонентом любой программной платформы, предназначенной для исследования операций, поскольку именно язык моделирования позволяет описать оптимизируемую модель наиболее естественным образом. Иными словами, язык моделирования должен быть удобен для исследователя и понятен

для компьютера. О расчетах в исследовании операций с использованием языков Julia и JuMP рассказано в [2]. Акцент сделан на математическую оптимизацию. В частности, авторы продемонстрировали возможности алгебраического моделирования для линейной и нелинейной оптимизации. Подтверждена высокая эффективность вычислений программ на языке Julia.

Линейное программирование является одним из ключевых инструментов исследования операций. Оно предназначено для расчета значений переменных, которые максимизируют линейную функцию с учетом имеющихся линейных ограничений. Именно эта фундаментальная проблема, а также алгоритмы ее решения составляют большую часть вычислений, связанных с исследованиями операций. В [2] приведены результаты анализа достижений в языках программирования, которые влияют на реализацию алгоритмов исследования операций. В качестве иллюстрации использованы задачи, решаемые с применением линейного и нелинейного программирования.

В статье [8] описана библиотека выпуклого оптимизационного моделирования Convex, задача которой заключается в переводе проблемы, представленной на понятном исследователю языке, в абстрактное синтаксическое дерево, описывающее проблему, что позволяет Convex проверить, выполняются ли правила дисциплинированного выпуклого программирования и передать решение проблемы подходящему решателю. Эти операции выполняются в Julia с использованием множественной диспетчеризации. Выбор решателя осуществляется в Convex автоматически. Показано, что задачу математической оптимизации можно из удобной для чтения человеком формы преобразовать в абстрактное синтаксическое дерево, представляющее эту задачу, с использованием набора функций. Например, Convex может проверить, является ли задача выпуклой, применяя правила дисциплинированного выпуклого программирования [9].

В работе [10] приведена программная платформа StochasticPrograms.jl с открытым исходным кодом, которая написана на языке Julia и предназначена для стохастического программирования. Платформа включает в себя инструменты для моделирования и алгоритмы оптимизации. Показано, что платформу можно использовать для крупномасштабных распределенных вычислений, применяя для расчетов как свободно распространяемые, так и коммерческие решатели.

Библиотека DifferentialEquations.jl для решения дифференциальных уравнений в Julia описана в [11]. Перечень решаемых с ее помощью задач включает, в частности, обыкновенные, стохастические, алгебраические и

гибридные дифференциальные уравнения, а также дифференциальные уравнения в частных производных. Библиотека DifferentialEquations.jl предоставляет обобщенный пользовательский интерфейс для решения и анализа различных форм дифференциальных уравнений, обеспечивая при этом высокую производительность вычислений. Авторы обращают внимание на такие особенности библиотеки, как удобство реализации параллельных вычислений, возможность выполнения вычислений с повышенной точностью и символическое вычисление якобианов.

Дифференциальные уравнения являются важной составной частью многих научных моделей, в частности, с их помощью удается описать такие крупномасштабные физические явления, как климат Земли и планетные системы, химические и биохимические реакции и т. д. Наличие библиотеки DifferentialEquations.jl дает возможность исследователям выполнять отработку алгоритмов и реализовывать крупномасштабные модели в Julia, не прибегая к использованию двух языков, когда требуется высокая производительность вычислений. Унификация пользовательского интерфейса библиотеки DifferentialEquations.jl для решения дифференциальных уравнений различных типов реализуется за счет множественной диспетчеризации.

Алгоритм метода расщепления конических операторов (COSMO) для квадратичной целевой функции и конических ограничений, а также его свободно распространяемая программная реализация на языке Julia представлены в [12]. Особенности алгоритма позволяют использовать его для решения задач оптимизации большой размерности, например, полуопределенных программ оптимизации портфеля, теории графов и робастного управления. Оператор COSMO позволяет решать задачи оптимизации для квадратичных целевых функций, не прибегая к переформулировке проблемы. Библиотека написана с использованием модулей, что облегчает ее расширение и тестирование новых версий. При разработке библиотеки использован объектно-ориентированный подход.

В статье [13] приводится описание RAFF.jl — открытой библиотеки на языке Julia, предназначенной для робастной аппроксимации экспериментальных данных. Особенности алгоритма, реализованного в библиотеке, позволяют определять возможные ошибочные данные (выбросы), благодаря чему удается уменьшить погрешность аппроксимации. Задачи линейной и нелинейной регрессии часто встречаются во многих областях науки. Массив экспериментальных данных может содержать ошибочные значения, которые оказывают существенное влияние на результаты аппроксимации, если использовать их в расчетах параметров регрессии.

Библиотека RAFF.jl позволяет автоматически определять выбросы, используя систему голосования.

Modelica — специализированный язык, предназначенный для компонентно-ориентированного моделирования сложных систем, в частности, для моделирования тепловых сетей. Опыт показывает, что возможностей языка Modelica зачастую бывает недостаточно для учета всех деталей реального процесса. В [14] рассказано о расширении возможностей моделирования с компонентом, созданным с использованием языка программирования Julia. Модель среды состоит из структуры данных и набора функций, предназначенных для обработки этих данных. Свойства жидкости вычисляются с использованием набора переменных, которые характеризуют термодинамическое состояние среды. Например, термодинамические свойства влажного воздуха в состоянии идеального газа, описываемые моделью Modelica.Media.Air.MoistAir, можно рассчитать по заданным значениям температуры, давления и массовой доли воды. Авторы статьи ставили перед собой задачу разработать модель и использовать ее для моделирования процессов в тепловых сетях, в частности, теплообменников, систем кондиционирования воздуха, колонн дистилляции и электростанций. Компонент Modelica.Fluid предназначен для обеспечения гибкости моделей за счет разделения моделей жидкости и среды, что позволяет использовать модель трубы со средой, которая имеет разные термодинамические состояния.

Большая статья [15] посвящена описанию учебной библиотеки для проведения гидродинамических расчетов на языке Julia. Рассмотрены различные концепции, связанные с пространственной и временной дискретизациями, явными и неявными числовыми схемами, многошаговыми числовыми схемами, численными методами и итерационными решателями в вычислительной гидродинамике. Эти концепции проиллюстрированы примерами расчетов с использованием линейного уравнения конвекции, невязкого уравнения Бюргерса и двумерного уравнения Пуассона. Приведены результаты сравнения производительности решателей уравнений Навье — Стокса, написанных на языках Python и Julia. Разработанная библиотека используется в процессе обучения студентов старших курсов методам вычислительной гидродинамики. Исходные тексты библиотеки на языке Julia с примерами учебных задач можно найти по адресу github.com/surajp92/CFD_Julia.

JuliaFEM — библиотека с открытым исходным кодом на языке Julia (www.juliafem.org), предназначенная для проведения параллельных расчетов методом конечного элемента на компьютерном кластере [16]. При

проектировании библиотеки была предусмотрена возможность масштабирования кластера до тысяч компьютеров. Основной принцип проектирования — все нелинейно. При разработке JuliaFEM использована библиотека автоматического дифференцирования ForwardDi [17] для расчета градиентов, якобианов и гессианов. Библиотека обеспечивает взаимодействие с CODE ASTER — еще одним МКЭ-решателем с открытым исходным кодом (www.code-aster.org). Пример использования библиотеки JuliaFEM для модельного расчета собственных частот кронштейна приведен в [18].

В работе [19] рассказано о написанной на языке Julia библиотеке GaussianProcesses.jl, предназначенной для численного моделирования гауссовых процессов. Для оптимизации параметров процесса использована библиотека Optim.jl [6], которая предоставляет несколько эффективных алгоритмов оптимизации без ограничений. Априорные распределения для гиперпараметров можно задать с использованием библиотеки Distributions.jl [20], которая является частью экосистемы JuliaStats (juliastats.org).

С проблемой двух языков приходится сталкиваться и разработчикам приложений в области робототехники. В [21] показано, как можно решить эту проблему с использованием языка Julia при создании программы управления роботом, который может балансировать на плоской поверхности. Представлено несколько разработанных программ на языке Julia.

Вероятностное программирование, которое используется в системах вероятностного машинного обучения, требует гибких и эффективных машин вывода. В [22] приведена система Turing (turing.ml/dev/), предназначенная для гибкого вероятностного программирования. Синтаксис Turing достаточно прост, при этом предоставляется возможность использования нескольких алгоритмов вывода. Система реализована в виде библиотеки на языке Julia, обеспечивается поддержка наиболее распространенных методов Монте-Карло.

Численное моделирование открытых квантовых систем имеет важное значение для исследований в таких областях, как квантовая оптика или квантовая информация, поскольку число аналитически разрешимых систем весьма ограничено. В [23] приведена программная платформа с открытым исходным кодом QuantumOptics.jl, предназначенная для эффективного численного моделирования открытых квантовых систем. Аналогом данной платформы можно считать разработанный более 20 лет назад Quantum Optics (QO) Toolbox для MATLAB [24]. Сравнительно недавно была разработана программная платформа с открытым исходным кодом

QuTiP (Quantum Toolbox in Python) на языке Python [25]. Поскольку расчеты на языке Python выполняются довольно медленно, для выполнения трудоемких вычислений в среде QuTiP необходимо подключать библиотеки, написанные на C/C++ или Fortran, что не всегда удобно. Написанная на языке Julia платформа QuantumOptics.jl лишена подобного недостатка.

Библиотека Julia для численных вычислений в теории квантовой информации QuantumInformation.jl приведена в [26]. Назначение библиотеки — предоставление функций для создания квантовых состояний, манипулирования ими с помощью квантовых каналов, расчета функционалов на этих объектах и выборки их случайным образом из различных распределений.

Две новых библиотеки для решения задач из области компьютерной алгебры (Nemo) и алгебраической теории чисел (Hecke) на языке Julia приведены в [27]. Продемонстрирована высокая скорость вычислений, выполняемых с использованием этих библиотек. Если необходимо, то можно использовать предоставляемый языком Julia интерфейс к известным библиотекам, написанным на таких языках C/C++, как Flint, Arb, Antic и Singular. Приведены примеры использования библиотек Nemo и Hecke, алгоритмы, которые реализованы при их разработке.

Работа [28] посвящена описанию возможностей применения языка Julia в астрономии и астрофизике. Организация JuliaAstro GitHub (github.com/JuliaAstro) собирает все библиотеки на языке Julia, предназначенные для решения задач из области астрономии. С точки зрения авторов статьи, язык Julia целесообразно использовать в астрофизике по крайней мере в двух случаях.

Для анализа больших массивов данных, когда существует необходимость проведения интенсивных вычислений (например, в проекте Celeste приходится иметь дело с объемом данных порядка 178 ТБ из каталога SDSS).

Существующие программы зачастую трудно использовать в интерактивном режиме, язык Julia позволяет осуществлять это без проблем либо через командную строку, либо через оболочку Jupiter.

В работе [29] представлена программная платформа Yao.jl, написанная на языке Julia, которая предназначена для разработки квантовых алгоритмов. Авторы утверждают, что их платформа отличается высокой производительностью при моделировании небольших квантовых цепей. Описаны принципы проектирования и некоторые важные технологические подробности Yao.jl.

Сравнительный анализ производительности вычислений при решении задач механики твердого тела методом материальной точки с использова-

нием языков программирования Julia и MATLAB приведен в [30]. Отмечено, что оба языка позволяют достаточно быстро создавать модель, однако расчеты с использованием Julia выполняются быстрее, чем в среде MATLAB. Авторы публикуют тексты разработанных ими программ расчета столкновения двух одинаковых упругих дисков, деформации консольной балки, двумерного моделирования ударного взаимодействия стального диска и алюминиевой мишени.

Библиотека для оформления научных статей Weave.jl с использованием Julia описана в [31]. Она позволяет записывать в один документ текст, математические выражения и программный код, которые затем преобразуются в отчет. Предусмотрена возможность использования нескольких языков разметки, графиков, построенных средствами специализированных библиотек Julia, и других объектов, отображаемых средствами языка Juila. Библиотека предназначена для оформления научных статей, учебных материалов, пособий и т. д.

**Учебники по математике.** Язык программирования Julia разработан в MIT (Massachusetts Institute of Technology), поэтому именно университеты США (в частности MIT, Stanford University) наиболее активно внедряют этот язык в учебный процесс. Недавно появилось несколько учебников по математике, теоретический материал которых сопровождается программами на языке Julia. Эти учебники тоже включены в настоящий небольшой обзор.

В свободно распространяемом (web.stanford.edu/~boyd/vmls/) учебнике по прикладной линейной алгебре [32] рассмотрены традиционные вопросы из курса линейной алгебры: понятия нормы и расстояния, работа с матрицами и векторами, линейные уравнения, умножение и обращение матриц, а также некоторые прикладные аспекты применения аппарата линейной алгебры: линейный и нелинейный метод наименьших квадратов, с ограничениями и без, линейные динамические системы. В Приложение к [32] включены дополнительные материалы, которые, по сути дела, заменяют семинарские занятия по прикладной линейной алгебре. Приложение содержит задачи по курсу линейной алгебры и примеры решения этих задач на языке программирования Julia с использованием математического аппарата, изложенного в учебнике. Для решения задач используются как библиотеки, которые являются частью дистрибутива Julia, так и дополнительные библиотеки.

В [33] подробно рассмотрены основные алгоритмы оптимизации: методы нулевого, первого и второго порядков, стохастические, популяционные, методы оптимизации с линейными ограничениями. Рассмотрены

также некоторые модели: суррогатные и вероятностно-суррогатные, вопросы оптимизации в условиях неопределенности и распространения погрешности. Практически все алгоритмы реализованы на языке Julia, что позволяет студенту получить не только знания, но и умения применять их на практике.

Недавно вышло второе издание учебника по исследованию операций с использованием языка Julia [34]. С электронной версией первого издания можно ознакомиться по ссылке www.softcover.io/read/7b8eb7d0/juliabook. Рассмотрены основы языка Julia, некоторые аспекты численных методов, симплекс-метод, линейная оптимизация, общие задачи оптимизации, методы Монте-Карло, некоторые полезные библиотеки на языке Julia.

В свободном доступе также можно найти предварительный вариант учебника по основам наук о данных, машинному обучению и искусственному интеллекту [35], теоретические материалы которого иллюстрируются текстами программ на языке Julia, в учебнике рассмотрены вопросы обработки данных, линейной регрессии, проверки гипотез, основы машинного обучения, моделирования динамических моделей.

К сожалению, учебников по языку Julia на русском языке пока немного. Можно рекомендовать лишь пособие [36], изданный на русском языке перевод книги [37], к сожалению, морально устарел.

**Заключение.** Рассмотрены некоторые прикладные библиотеки языка программирования Julia, предназначенные для решения научных и технических задач.

Проведенный анализ свидетельствует о целесообразности использования языка программирования Julia в учебном процессе при подготовке специалистов инженерных и естественно-научных специальностей.

## ЛИТЕРАТУРА

[1] Ayer V.M., Miguez S., Toby B.H. Why scientists should learn to program in Python. *Powder Diffr.*, 2014, vol. 29, no. s2, pp. S48–S64.
DOI: https://doi.org/10.1017/S0885715614000931

[2] Lubin M., Dunning I. Computing in operations research using Julia. *INFORMS J. Comput.*, 2015, vol. 27, no. 2, pp. 193–430. DOI: https://doi.org/10.1287/ijoc.2014.0623

[3] Bezanson J., Karpinsky S., Shah V.B., et al. Julia: a fast dynamic language for technical computing. *arXiv.org: веб-сайт.* URL: https://arxiv.org/abs/1209.5145 (дата обращения: 19.12.2019).

[4] Bezanson J., Edelman A., Karpinsky S., et al. Julia: a fresh approach to numerical computing. *SIAM Rev.*, 2017, vol. 59, no. 1, pp. 65–98.
DOI: https://doi.org/10.1137/141000671

[5] Lattner C., Adve V. LLVM: A compilation framework for lifelong program analysis & transformation. *Proc. CGO Int. Symp.*, 2004, pp. 75–86.
DOI: https://doi.org/10.1109/CGO.2004.1281665

[6] Mogensen P.K., Riseth A.N. Optim: a mathematical optimization package for Julia. *J. Open Source Softw.*, 2018, vol. 3, no. 24. DOI: https://doi.org/10.21105/joss.00615

[7] Dunning I., Huchette J., Lubin M. JuMP: a modeling language for mathematical optimization. *SIAM Rev.*, 2017, vol. 59, no. 2, pp. 295–320.
DOI: https://doi.org/10.1137/15M1020575

[8] Udell M., Mohan K., Zeng D., et al. Convex optimization in Julia. *Proc. 1st Workshop for High Performance Technical Computing in Dynamic Languages*, 2014, pp. 18–28.
DOI: https://doi.org/10.1109/HPTCDL.2014.5

[9] Grant M., Boyd S., Ye Y. Disciplined convex programming. In: Global optimization. Springer, 2006, pp. 155–210.

[10] Biel M., Johansson M. Efficient stochastic programming in Julia. *arXiv.org: веб-сайт.* URL: https://arxiv.org/abs/1909.10451 (дата обращения: 19.12.2019).

[11] Rackauckas C., Nie Q. DifferentialEquations.jl — a performant and feature-rich ecosystem for solving differential equations in Julia. *J. Open Res. Softw.*, 2017, vol. 5, no. 1, art. 15. DOI: https://doi.org/10.5334/jors.151

[12] Garstka M., Cannon M., Goulart P. COSMO: a conic operator splitting method for convex conic problems. *arXiv.org: веб-сайт.*
URL: https://arxiv.org/abs/1901.10887 (дата обращения: 19.12.2019).

[13] Castelani E.V., Lopes R., Wesley V.I., et al. RAFF.jl: robust algebraic fitting function in Julia. *J. Open Res. Softw.,* 2019, vol. 4, no. 39.
DOI: https://doi.org/10.21105/joss.01385

[14] Otter M., Elmqvist H., Zimmer D., et al. Thermodynamic property and fluid modeling with modern programming language construct. *Proc. 13th Int. Modelica Conf.* Regensburg, Germany, 2019, pp. 589–598. DOI: https://doi.org/10.3384/ecp19157589

[15] Pawar S., San O. CFD Julia: a learning module structuring an introductory course on computational fluid dynamics. *Fluids*, 2019, vol. 4, no. 3, art. 159.
DOI: https://doi.org/10.3390/fluids4030159

[16] Frondelius T., Aho J. JuliaFEM — open source solver for both industrial and academia usage. *Rakenteiden Mekaniikka*, 2017, vol. 50, no. 3, pp. 229–233.
DOI: https://doi.org/10.23998/rm.64224

[17] Revels J., Lubin M., Papamarkou T. Forward-mode automatic differentiation in Julia. *arXiv.org: веб-сайт.* URL: https://arxiv.org/abs/1607.07892 (дата обращения: 19.12.2019).

[18] Rapo M., Aho J., Frondelius T. Natural frequency calculations with JuliaFEM. *Rakenteiden Mekaniikka*, 2017, vol. 50, no. 3, pp. 300–303.
DOI: https://doi.org/10.23998/rm.65040

[19] Fairbrother J., Nemeth C., Rischard M., et al. GaussianProcesses.jl: a Nonparametric Bayes package for the Julia language. *arXiv.org: веб-сайт.*
URL: https://arxiv.org/abs/1812.09064 (дата обращения: 19.12.2019).

[20] Besançon M., Anthoff D., Arslan A., et al. Distributions.jl: definition and modeling of probability distributions in the JuliaStats ecosystem. *arXiv.org: веб-сайт.*
URL: https://arxiv.org/abs/1907.08611 (дата обращения: 19.12.2019).

[21] Koolen T., Deits R. Julia for robotics: simulation and real-time control in a high-level programming language. *Proc. ICRA,* 2019.
DOI: https://doi.org/10.1109/ICRA.2019.8793875

[22] Ge H., Xu K., Ghahramani Z. Turing: A language for flexible probabilistic inference. *Proc. 21st Int. Conf. on Artificial Intelligence and Statistics.* Vol. 4. 2018, pp. 1682–1690.

[23] Krämer S., Plankensteiner D., Ostermann L., et al. QuantumOptics.jl: a Julia framework for simulating open quantum systems. *Comput. Phys. Commun.*, 2018, vol. 227, pp. 109–116. DOI: https://doi.org/10.1016/j.cpc.2018.02.004

[24] Tan S.M. A computational toolbox for quantum and atomic optics. *J. Opt. B: Quantum Semiclass. Opt.*, 1999, vol. 1, no. 4, art. 424.
DOI: https://doi.org/10.1088/1464-04266/1/4/312

[25] Johansson J.R., Nation P.D., Nori F. QuTiP: an open-source Python framework for the dynamics of open quantum systems. *Comput. Phys. Commun.*, 2012, vol. 183, no. 8, pp. 1760–1772. DOI: https://doi.org/10.1016/j.cpc.2012.02.021

[26] Gawron P., Kurzyk D., Pawela L. QuantumInformation.jl — a Julia package for numerical computation in quantum information theory. *PlOS ONE*, 2018, vol. 13, no. 12, art. e0209358. DOI: https://doi.org/10.1371/journal.pone.0209358

[27] Fieker C., Hart W., Hofmann T., et al. Nemo/Hecke: computer algebra and number theory packages for the Julia programming language. *Proc. ACM ISAAC,* 2017, pp. 157–164. DOI: https://doi.org/10.1145/3087604.3087611

[28] Tomasi M., Giordano M. Towards new solutions for scientific computing: the case of Julia. *arXiv.org: веб-сайт.* URL: https://arxiv.org/abs/1812.01219 (дата обращения: 19.12.2019).

[29] Luo X.Z., Liu J.G., Zhang P., et al. Yao.jl: extensible, efficient framework for quantum algorithm design. *arXiv.org: веб-сайт.*
URL: https://arxiv.org/abs/1912.10877 (дата обращения: 19.12.2019).

[30] Sinaie S., Nguyen V.P., Nguyen C.T., et al. Programming the material point method in Julia. *Adv. Eng. Softw.,* 2017, vol. 105, pp. 17–29.
DOI: https://doi.org/10.1016/j.advengsoft.2017.01.008

[31] Pastell M. Weave.jl: scientific reports using Julia. *J. Open Source Softw.*, 2017, vol. 2, no. 11, art. 204. DOI: https://doi.org/10.21105/joss.00204

[32] Boyd S., Vandenberghe L. Introduction to applied linear algebra: vectors, matrices, and least squares. Cambridge University Press, 2018.

[33] Kochenderfer M.J., Wheeler T.A. Algorithms for optimization. MIT Press, 2019.

[34] Kwon C. Julia programming for operations research: a primer on computing. Independently Publ., 2019.

[35] Klok H., Nazarathy Y. Statistics with Julia: fundamentals for data science, machine learning and artificial intelligence.
URL: https://github.com/h-Klok/StatsWithJuliaBook (дата обращения: 19.12.2019).

[36] Антонюк В.А. Язык Julia как инструмент исследователя. М., Изд-во МГУ им. М.В. Ломоносова, 2019.

[37] Sherrington M. Mastering Julia. Birminghan, Packt Publishing Ltd., 2015.

**Белов Глеб Витальевич** — д-р техн. наук, ведущий научный сотрудник лаборатории химической термодинамики кафедры «Физическая химия» химического факультета МГУ им. М.В. Ломоносова (Российская Федерация, 119991, Москва, Ленинские горы, д. 1, стр. 3).

**Аристова Нина Михайловна** — канд. хим. наук, старший научный сотрудник лаборатории теплофизических баз данных отдела физики экстремальных состояний ОИВТ РАН (Российская Федерация, 123412, Москва, Ижорская ул., д. 13, стр. 2).

**Просьба ссылаться на эту статью следующим образом:**

Белов Г.В., Аристова Н.М. О возможностях использования языка программирования Julia для решения научных и технических задач. *Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение*, 2020, № 2, с. 27–43.
DOI: https://doi.org/10.18698/0236-3933-2020-2-27-43

## ON THE POTENTIAL OF THE JULIA PROGRAMMING LANGUAGE FOR SOLVING SCIENTIFIC AND ENGINEERING PROBLEMS

**G.V. Belov**[1] gbelov@yandex.ru
**N.M. Aristova**[2] aristo2012@yandex.ru

[1] **Lomonosov Moscow State University, Moscow, Russian Federation**
[2] **Joint Institute for High Temperatures, Russian Academy of Sciences, Moscow, Russian Federation**

**Abstract**

The paper provides a brief description of the Julia language and discusses several libraries and software packages written in this language and designed to solve various scientific and engineering problems. The Julia language has been developed relatively recently, but our analysis of existing publications indicates its growing popularity. We limited the scope of the paper to review only those few libraries and software packages that may be useful to researchers and engineers who develop software simulations and conduct numerical experiments.

**Keywords**

*Julia programming language, mathematical simulation, software library*

Specifically, we did not consider dedicated libraries for economic, biological and medical applications. At present, there exists a large number of programming languages, each of them developed to overcome the limitations of the ones already available. Each language has its advantages and disadvantages and is suited best for a particular purpose. The objective of our paper is to familiarise the reader with the potential of the Julia programming language for solving scientific and engineering problems

Received 06.02.2020
Accepted 10.02.2020
© Author(s), 2020

## REFERENCES

[1] Ayer V.M., Miguez S., Toby B.H. Why scientists should learn to program in Python. *Powder Diffr.*, 2014, vol. 29, no. s2, pp. S48–S64.
DOI: https://doi.org/10.1017/S0885715614000931

[2] Lubin M., Dunning I. Computing in operations research using Julia. *INFORMS J. Comput.*, 2015, vol. 27, no. 2, pp. 193–430. DOI: https://doi.org/10.1287/ijoc.2014.0623

[3] Bezanson J., Karpinsky S., Shah V.B., et al. Julia: a fast dynamic language for technical computing. *arXiv.org: website.* Available at: https://arxiv.org/abs/1209.5145 (accessed: 19.12.2019).

[4] Bezanson J., Edelman A., Karpinsky S., et al. Julia: a fresh approach to numerical computing. *SIAM Rev.*, 2017, vol. 59, no. 1, pp. 65–98.
DOI: https://doi.org/10.1137/141000671

[5] Lattner C., Adve V. LLVM: A compilation framework for lifelong program analysis & transformation. *Proc. CGO Int. Symp.*, 2004, pp. 75–86.
DOI: https://doi.org/10.1109/CGO.2004.1281665

[6] Mogensen P.K., Riseth A.N. Optim: a mathematical optimization package for Julia. *J. Open Source Softw.*, 2018, vol. 3, no. 24. DOI: https://doi.org/10.21105/joss.00615

[7] Dunning I., Huchette J., Lubin M. JuMP: a modeling language for mathematical optimization. *SIAM Rev.*, 2017, vol. 59, no. 2, pp. 295–320.
DOI: https://doi.org/10.1137/15M1020575

[8] Udell M., Mohan K., Zeng D., et al. Convex optimization in Julia. *Proc. 1st Workshop for High Performance Technical Computing in Dynamic Languages*, 2014, pp. 18–28. DOI: https://doi.org/10.1109/HPTCDL.2014.5

[9] Grant M., Boyd S., Ye Y. Disciplined convex programming. In: Global optimization. Springer, 2006, pp. 155–210.

[10] Biel M., Johansson M. Efficient stochastic programming in Julia. *arXiv.org: website.* Available at: https://arxiv.org/abs/1909.10451 (accessed: 19.12.2019).

[11] Rackauckas C., Nie Q. DifferentialEquations.jl — a performant and feature-rich ecosystem for solving differential equations in Julia. *J. Open Res. Softw.*, 2017, vol. 5, no. 1, art. 15. DOI: https://doi.org/10.5334/jors.151

[12] Garstka M., Cannon M., Goulart P. COSMO: a conic operator splitting method for convex conic problems. *arXiv.org: website.*
Available at: https://arxiv.org/abs/1901.10887 (accessed: 19.12.2019).

[13] Castelani E.V., Lopes R., Wesley V.I., et al. RAFF.jl: robust algebraic fitting function in Julia. *J. Open Res. Softw.*, 2019, vol. 4, no. 39. DOI: https://doi.org/10.21105/joss.01385

[14] Otter M., Elmqvist H., Zimmer D., et al. Thermodynamic property and fluid modeling with modern programming language construct. *Proc. 13th Int. Modelica Conf.* Regensburg, Germany, 2019, pp. 589–598. DOI: https://doi.org/10.3384/ecp19157589

[15] Pawar S., San O. CFD Julia: a learning module structuring an introductory course on computational fluid dynamics. *Fluids*, 2019, vol. 4, no. 3, art. 159.
DOI: https://doi.org/10.3390/fluids4030159

[16] Frondelius T., Aho J. JuliaFEM-open source solver for both industrial and academia usage. *Rakenteiden Mekaniikka*, 2017, vol. 50, no. 3, pp. 229–233.
DOI: https://doi.org/10.23998/rm.64224

[17] Revels J., Lubin M., Papamarkou T. Forward-mode automatic differentiation in Julia. *arXiv.org: website.* Available at: https://arxiv.org/abs/1607.07892 (accessed: 19.12.2019).

[18] Rapo M., Aho J., Frondelius T. Natural frequency calculations with JuliaFEM.
*Rakenteiden Mekaniikka*, 2017, vol. 50, no. 3, pp. 300–303.
DOI: https://doi.org/10.23998/rm.65040

[19] Fairbrother J., Nemeth C., Rischard. M., et al. GaussianProcesses.jl: a Nonparametric Bayes package for the Julia language. *arXiv.org: website.*
Available at: https://arxiv.org/abs/1812.09064 (accessed: 19.12.2019).

[20] Besançon M., Anthoff D., Arslan A., et al. Distributions.jl: definition and modeling of probability distributions in the JuliaStats ecosystem. *arXiv.org: website.*
Available at: https://arxiv.org/abs/1907.08611 (accessed: 19.12.2019).

[21] Koolen T., Deits R. Julia for robotics: simulation and real-time control in a high-level programming language. *Proc. ICRA*, 2019.
DOI: https://doi.org/10.1109/ICRA.2019.8793875

[22] Ge H., Xu K., Ghahramani Z. Turing: A language for flexible probabilistic inference. *Proc. 21st Int. Conf. on Artificial Intelligence and Statistics.* Vol. 4. 2018, pp. 1682–1690.

[23] Krämer S., Plankensteiner D., Ostermann L., et al. QuantumOptics.jl: a Julia framework for simulating open quantum systems. *Comput. Phys. Commun.*, 2018, vol. 227, pp. 109–116. DOI: https://doi.org/10.1016/j.cpc.2018.02.004

[24] Tan S.M. A computational toolbox for quantum and atomic optics. *J. Opt. B: Quantum Semiclass. Opt.*, 1999, vol. 1, no. 4, art. 424.
DOI: https://doi.org/10.1088/1464-04266/1/4/312

[25] Johansson J.R., Nation P.D., Nori F. QuTiP: an open-source Python framework for the dynamics of open quantum systems. *Comput. Phys. Commun.*, 2012, vol. 183, no. 8, pp. 1760–1772. DOI: https://doi.org/10.1016/j.cpc.2012.02.021

[26] Gawron P., Kurzyk D., Pawela L. QuantumInformation.jl — a Julia package for numerical computation in quantum information theory. *PlOS ONE*, 2018, vol. 13, no. 12, art. e0209358. DOI: https://doi.org/10.1371/journal.pone.0209358

[27] Fieker C., Hart W., Hofmann T., et al. Nemo/Hecke: computer algebra and number theory packages for the Julia programming language. *Proc. ACM ISAAC*, 2017, pp. 157–164. DOI: https://doi.org/10.1145/3087604.3087611

[28] Tomasi M., Giordano M. Towards new solutions for scientific computing: the case of Julia. *arXiv.org: website.*
Available at: https://arxiv.org/abs/1812.01219 (accessed: 19.12.2019).

[29] Luo X.Z., Liu J.G., Zhang P., et al. Yao.jl: extensible, efficient framework for quantum algorithm design. *arXiv.org: website.*
Available at: https://arxiv.org/abs/1912.10877 (accessed: 19.12.2019).

[30] Sinaie S., Nguyen V.P., Nguyen C.T., et al. Programming the material point method in Julia. *Adv. Eng. Softw.*, 2017, vol. 105, pp. 17–29.
DOI: https://doi.org/10.1016/j.advengsoft.2017.01.008

[31] Pastell M. Weave.jl: scientific reports using Julia. *J. Open Source Softw.*, 2017, vol. 2, no. 11, art. 204. DOI: https://doi.org/10.21105/joss.00204

[32] Boyd S., Vandenberghe L. Introduction to applied linear algebra: vectors, matrices, and least squares. Cambridge University Press, 2018.

[33] Kochenderfer M.J., Wheeler T.A. Algorithms for optimization. MIT Press, 2019.

[34] Kwon C. Julia programming for operations research: a primer on computing. Independently Publ., 2019.

[35] Klok H., Nazarathy Y. Statistics with Julia: fundamentals for data science, machine learning and artificial intelligence.
Available at: https://github.com/h-Klok/StatsWithJuliaBook (accessed: 19.12.2019).

[36] Antonyuk V.A. Yazyk Julia kak instrument issledovatelya [Julia language as a tool of explorer]. Moscow, Lomonosov MSU Publ., 2019.

[37] Sherrington M. Mastering Julia. Birminghan, Packt Publishing Ltd., 2015.

**Belov G.V.** — Dr. Sc. (Eng.), Leading Research Fellow, Laboratory of Chemical Thermodynamics, Department of Physical Chemistry, Faculty of Chemistry, Lomonosov Moscow State University (Leninskie Gory 1, str. 3, Moscow, 119991 Russian Federation).

**Aristova N.M.** — Cand. Sc. (Chem.), Senior Research Fellow, Laboratory of Thermophysical Databases, Department of Extreme State Physics, Joint Institute for High Temperatures, Russian Academy of Sciences (Izhorskaya ul. 13, str. 2, Moscow, 125412 Russian Federation).

**Please cite this article in English as:**

Belov G.V., Aristova N.M. On the potential of the Julia programming language for solving scientific and engineering problems. *Herald of the Bauman Moscow State Technical University, Series Instrument Engineering*, 2020, no. 2, pp. 27–43 (in Russ.).
DOI: https://doi.org/10.18698/0236-3933-2020-2-27-43