

Introduction to the **julia** programming language

Bruno Amorim



Quantum Agora, 25th March 2021

Disclaimer:

I am a bad programmer (in python,C and Fortran)
I am slightly better with Julia

Why do we need another programming language?

Why do we need another programming language?



Easy languages to write code:

- Python
- Mathematica
- Matlab
- ...

Why do we need another programming language?

 **Easy** languages to write code:

- Python
- Mathematica
- Matlab
- ...

But slow! 

Why do we need another programming language?

 **Easy** languages to write code:

- Python
- Mathematica
- Matlab
- ...

But slow! 

 **Fast** languages:

- Fortran
- C
- C++
- ...

Why do we need another programming language?

 **Easy** languages to write code:

- Python
- Mathematica
- Matlab
- ...

But slow! 

 **Fast** languages:

- Fortran
- C
- C++
- ...

But hard! 

Why do we need another programming language?

 **Easy** languages to write code:

- Python
- Mathematica
- Matlab
- ...

But slow! 

 **Fast** languages:

- Fortran
- C
- C++
- ...

But hard! 

Compromise:

- 1) Prototype in easy language
- 2) When performance becomes critical, rewrite parts/whole code in fast language

Why do we need another programming language?

 **Easy** languages to write code:

- Python
- Mathematica
- Matlab
- ...

But slow! 

 **Fast** languages:

- Fortran
- C
- C++
- ...

But hard! 

Compromise:

- 1) Prototype in easy language
- 2) When performance becomes critical, rewrite parts/whole code in fast language

Problems:

- Duplication of effort
- Need to learn two languages or fast parts become a blackbox!

Why do we need another programming language?

😊 **Easy** languages to write code:

- Python
- Mathematica
- Matlab
- ...

But slow!

😊 **Fast** languages:

Fortran

Two language problem!

Compromise

- 1) Prototype in
- 2) When perform

fast language

Problems:

- Duplication of effort
- Need to learn two languages or fast parts become a blackbox!

Enter Julia

From <https://julialang.org/blog/2012/02/why-we-created-julia/>

“ Why We Created Julia



14 February 2012 | **Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman**

[Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman](#)

In short, because we are greedy.

”

From <https://julialang.org/blog/2012/02/why-we-created-julia/>

“ Why We Created Julia



14 February 2012 | **Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman**

[Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman](#)

”

In short, because we are greedy.

“ We want a language that's open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled.

(Did we mention it should be as fast as C?) ”

From <https://julialang.org/blog/2012/02/why-we-created-julia/>

“ Why We Created Julia



14 February 2012 | **Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman**

[Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman](#)

”

In short, because we are greedy.

“ We want a language that's open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled.

(Did we mention it should be as fast as C?) ”

From <https://julialang.org/blog/2012/02/why-we-created-julia/>

“ Why We Created Julia



14 February 2012 | **Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman**

[Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman](#)

In short, because we are greedy.

”

“ We want a language that's open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled.

(Did we mention it should be as fast as C?) ”

From <https://julialang.org/blog/2012/02/why-we-created-julia/>

“ Why We Created Julia



14 February 2012 | **Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman**

[Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman](#)

In short, because we are greedy.

”

“ We want a language that's open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled.

(Did we mention it should be as fast as C?) ”

From <https://julialang.org/blog/2012/02/why-we-created-julia/>

“ Why We Created Julia



14 February 2012 | **Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman**

[Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman](#)

In short, because we are greedy.

”

“ We want a language that's open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled.

(Did we mention it should be as fast as C?) ”

From <https://julialang.org/blog/2012/02/why-we-created-julia/>

“ Why We Created Julia



14 February 2012 | **Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman**

[Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman](#)

In short, because we are greedy.

”

“ We want a language that's open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled.

(Did we mention it should be as fast as C?) ”

From <https://julialang.org/blog/2012/02/why-we-created-julia/>

“ Why We Created Julia



14 February 2012 | **Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman**

[Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman](#)

”

In short, because we are greedy.

“ We want a language that's open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled.

(Did we mention it should be as fast as C?) ”

From <https://julialang.org/blog/2012/02/why-we-created-julia/>

“ Why We Created Julia



14 February 2012 | **Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman**

[Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman](#)

”

In short, because we are greedy.

“ We want a language that's open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled.

(Did we mention it should be as fast as C?) ”

From <https://julialang.org/blog/2012/02/why-we-created-julia/>

“ Why We Created Julia



14 February 2012 | **Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman**

[Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman](#)

In short, because we are greedy.

”

“ We want a language that's open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled.

(Did we mention it should be as fast as C?) ”

From <https://julialang.org/blog/2012/02/why-we-created-julia/>

“ Why We Created Julia



14 February 2012 | **Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman**

[Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman](#)

”

In short, because we are greedy.

“ We want a language that's open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled.

(Did we mention it should be as fast as C?) ”

So what is Julia?

So what is Julia?

New language: 1st public realease in 2012, stable 1.0 version in 2018

Current version: 1.6.0 (released yesterday!)

Initially development at MIT:

- *Julia: A Fast Dynamic Language for Technical Computing*, **arXiv:1209.5145**
Jeff Bezanson, Stefan Karpinski, Viral B. Shah, Alan Edelman
- *Julia: A Fresh Approach to Numerical Computing*, **arXiv:1411.1607**
Jeff Bezanson, Alan Edelman, Stefan Karpinski, Viral B. Shah
- *Abstraction in technical computing* (PhD Thesis)
<https://github.com/JeffBezanson/phdthesis/blob/master/main.pdf>
Jeff Bezanson

So what is Julia?

New language: 1st public realease in 2012, stable 1.0 version in 2018

Current version: 1.6.0 (released yesterday!)

Initially development at MIT:

- *Julia: A Fast Dynamic Language for Technical Computing*, **arXiv:1209.5145**
Jeff Bezanson, Stefan Karpinski, Viral B. Shah, Alan Edelman
- *Julia: A Fresh Approach to Numerical Computing*, **arXiv:1411.1607**
Jeff Bezanson, Alan Edelman, Stefan Karpinski, Viral B. Shah
- *Abstraction in technical computing* (PhD Thesis)
<https://github.com/JeffBezanson/phdthesis/blob/master/main.pdf>
Jeff Bezanson

- Open source
- Solves the two language problem
- Interactive
- High performance
- Simple syntax
- Aimed at scientific computing

Open source

source code hosted on github

Open source

source code hosted on github

The screenshot shows the GitHub repository page for JuliaLang/julia. The browser address bar displays 'github.com/JuliaLang/julia'. The repository name 'JuliaLang / julia' is prominently displayed at the top, accompanied by statistics: 33k stars, 4.4k forks, and 3.3k issues. Below this, navigation tabs include Code, Issues, Pull requests, Actions, Projects, Security, and Insights. The 'Code' tab is selected, showing a list of files and folders with their respective commit messages and timestamps. The 'About' section on the right provides information about the Julia Programming Language, including the website 'julialang.org/' and various tags like 'programming-language', 'science', 'machine-learning', 'hpc', 'julia', 'julia-language', 'scientific', 'hacktoberfest', and 'numerical'. The 'Releases' section shows the latest version 'v1.6.0' as the 'Latest' release, posted 6 hours ago, with a total of 104 releases.

GitHub - JuliaLang/julia

Why GitHub? Team Enterprise Explore Marketplace Pricing

Search Sign in Sign up

JuliaLang / julia

Sponsor Notifications Star 33k Fork 4.4k

<> Code Issues 3.3k Pull requests 848 Actions Projects 6 Security Insights

master 916 branches 104 tags Go to file Code

adomasbaliuka Fix manual claim that `run` returns `nothing` (#40155) d93fa40 7 hours ago 49,202 commits

.devcontainer	Add VS Code devcontainer definition (#34957)	13 months ago
.github	Set the `@JuliaLang/github-actions` team as the code owners for ...	4 months ago
base	add < to NamedTuple (#40147)	7 hours ago
cli	build: fix missing dependency links in Makefiles	16 days ago
contrib	Expand `Artifacts` tests for new platforms (#39829)	12 days ago
deps	[automated] Bump the Pkg stdlib from 7a9d9654 to af7e41cd (#400...	6 days ago
doc	Fix manual claim that `run` returns `nothing` (#40155)	7 hours ago
etc	fix write_base_cache to be consistend with the reader byte order (#...	3 years ago
src	fix #40050, handling fields that are pointers due to subtype circular...	yesterday
stdlib	Markdown: prevent display() error with empty list item (#40122)	yesterday

About

The Julia Programming Language

🔗 julialang.org/

programming-language science

machine-learning hpc julia

julia-language scientific

hacktoberfest numerical

Readme

View license

Releases 104

v1.6.0 Latest 6 hours ago

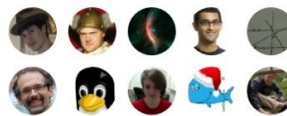
+ 103 releases

Open source

source code hosted on github

The screenshot shows the GitHub repository page for JuliaLang/julia. The repository has 33k stars and 4.4k forks. The main content area displays a list of recent commits, with the most recent one by adomasbaliuka titled 'Fix manual claim that `run` returns `nothing` (#40155)' committed 7 hours ago. The repository structure on the left includes folders like .devcontainer, .github, base, cli, contrib, deps, doc, etc, src, and stdlib. The right sidebar shows the repository's description as 'The Julia Programming Language' and a list of tags including programming-language, science, machine-learning, hpc, julia, julia-language, scientific, hacktoberfest, and numerical.

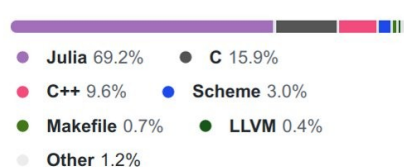
Contributors 1,101



Many contributors

+ 1,090 contributors

Languages



Julia is mostly written in Julia

This is true for most packages!

Solving the two language problem

Julia packages are mostly written in Julia

Solving the two language problem

Julia packages are mostly written in Julia

 [JuliaLinearAlgebra](#) / [IterativeSolvers.jl](#)

[Code](#) [Issues](#) 40 [Pull requests](#) 5 [Actions](#) [Projects](#) [Security](#) [Insights](#)

 [SciML](#) / [DifferentialEquations.jl](#)

[Code](#) [Issues](#) 110 [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#)

Languages



● **Julia** 100.0%

Languages



● **Julia** 96.4% ● **TeX** 3.6%

Solving the two language problem

Julia packages are mostly written in Julia

 [JuliaLinearAlgebra](#) / [IterativeSolvers.jl](#)

[Code](#) [Issues 40](#) [Pull requests 5](#) [Actions](#) [Projects](#) [Security](#) [Insights](#)

 [SciML](#) / [DifferentialEquations.jl](#)

[Code](#) [Issues 110](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#)

contrast with

 [numpy](#) / [numpy](#)

[Code](#) [Issues 2k](#) [Pull requests 235](#) [Actions](#) [Projects 8](#) [Wiki](#) [Security](#) [Insights](#)

 [scipy](#) / [scipy](#)

[Code](#) [Issues 1.3k](#) [Pull requests 319](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#)

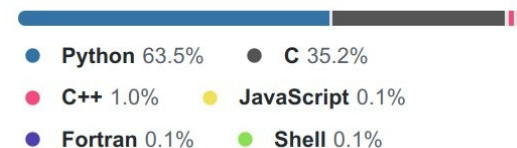
Languages



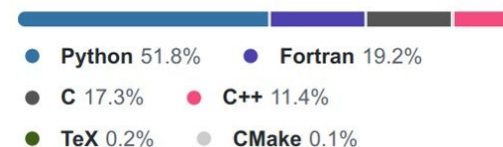
Languages



Languages



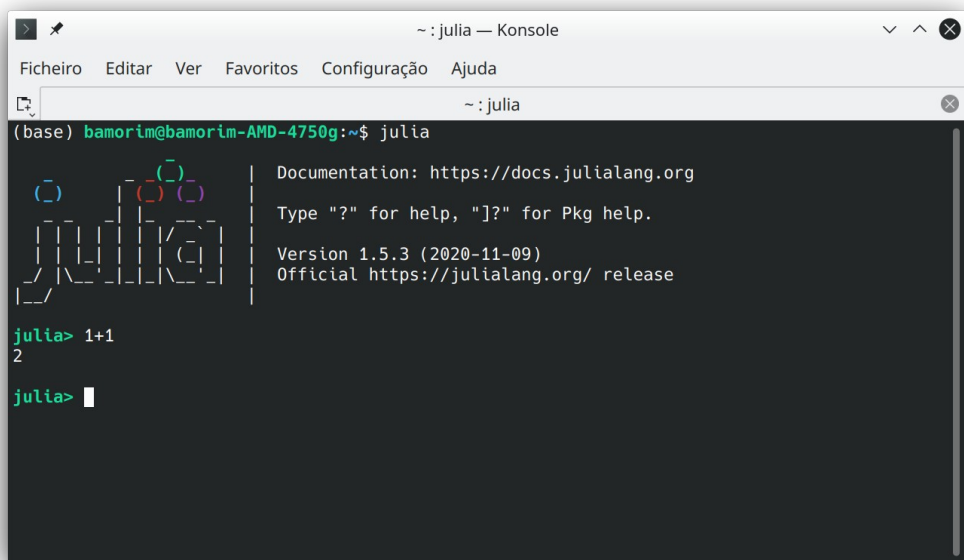
Languages



Julia is an interactive language

Julia is an interactive language

REPL (Read Eval Print Loop shell)



The screenshot shows a terminal window titled "~: julia — Konsole". The terminal displays the Julia REPL interface. At the top, there is a menu bar with options: "Ficheiro", "Editar", "Ver", "Favoritos", "Configuração", and "Ajuda". Below the menu bar, the terminal shows the prompt "(base) bamorim@bamorim-AMD-4750g:~\$ julia". The output of the command is a welcome message for Julia 1.5.3, including the documentation URL (<https://docs.julialang.org>), instructions on how to use help ("Type '?' for help, ']' for Pkg help."), the version number (1.5.3 (2020-11-09)), and the official website (<https://julialang.org/>). The terminal also shows a simple arithmetic calculation: "julia> 1+1" followed by the output "2". The prompt "julia>" is shown again at the bottom, indicating the REPL is ready for the next command.

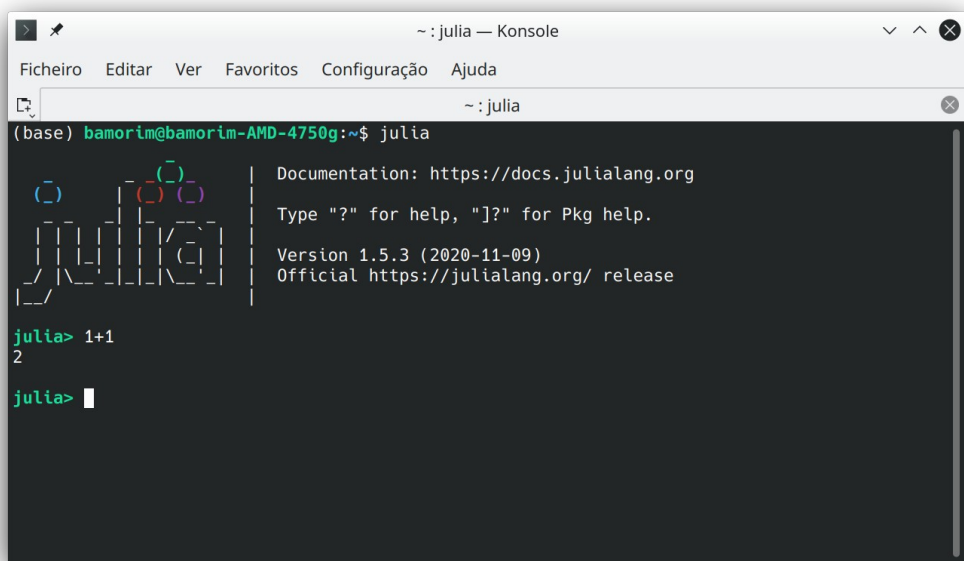
```
(base) bamorim@bamorim-AMD-4750g:~$ julia
Documentation: https://docs.julialang.org
Type "?" for help, "]" for Pkg help.
Version 1.5.3 (2020-11-09)
Official https://julialang.org/ release

julia> 1+1
2

julia>
```

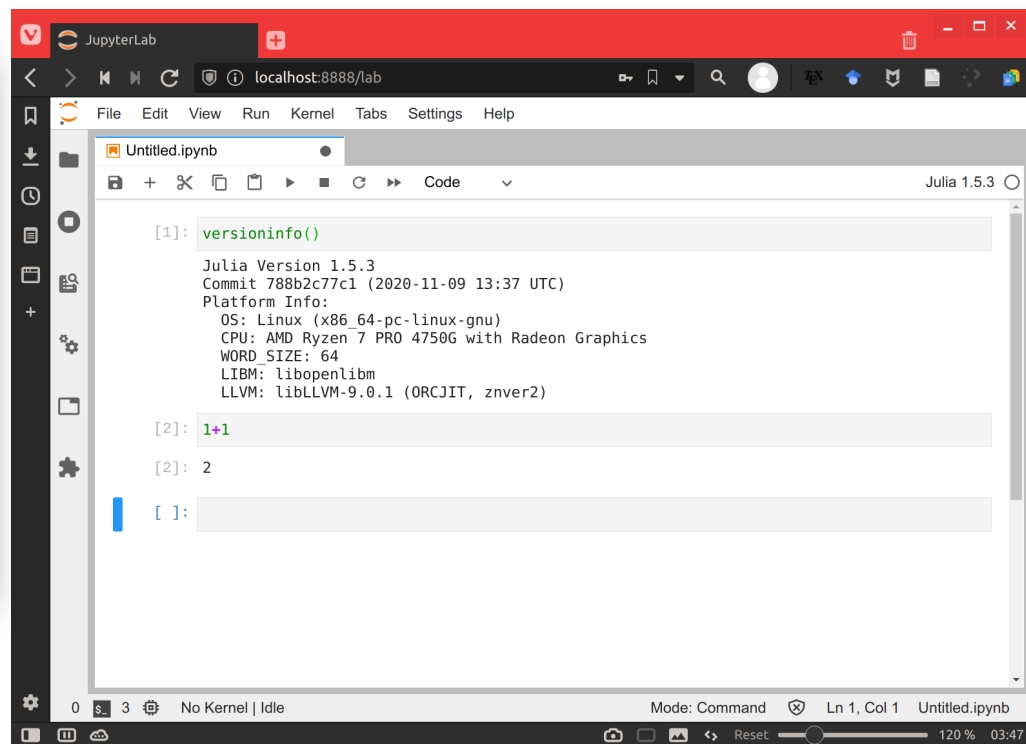
Julia is an interactive language

REPL (Read Eval Print Loop shell)

A terminal window titled '~: julia — Konsole' showing the Julia REPL. The prompt is '(base) bamorim@bamorim-AMD-4750g:~\$ julia'. The output shows the Julia logo, documentation link, version 1.5.3 (2020-11-09), and official website. Below this, the user enters 'julia> 1+1' and the output is '2'. The prompt 'julia>' is shown again.

```
~: julia — Konsole
Ficheiro  Editar  Ver  Favoritos  Configuração  Ajuda
~: julia
(base) bamorim@bamorim-AMD-4750g:~$ julia
Documentation: https://docs.julialang.org
Type "?" for help, "]" for Pkg help.
Version 1.5.3 (2020-11-09)
Official https://julialang.org/ release

julia> 1+1
2
julia> 
```

A screenshot of the JupyterLab interface. The top bar shows 'JupyterLab' and the URL 'localhost:8888/lab'. The main area displays a notebook titled 'Untitled.ipynb' with the following code and output:

```
[1]: versioninfo()
Julia Version 1.5.3
Commit 788b2c77c1 (2020-11-09 13:37 UTC)
Platform Info:
  OS: Linux (x86_64-pc-linux-gnu)
  CPU: AMD Ryzen 7 PRO 4750G with Radeon Graphics
  WORD_SIZE: 64
  LIBM: libopenlibm
  LLVM: libLLVM-9.0.1 (ORCJIT, znver2)

[2]: 1+1
[2]: 2

[ ]: 
```

The bottom status bar shows '0 3 No Kernel | Idle', 'Mode: Command', 'Ln 1, Col 1', 'Untitled.ipynb', and a zoom level of '120 %'.

Julia is an interactive language

REPL (Read Eval Print Loop shell)



```
~: julia — Konsole
Ficheiro  Editar  Ver  Favoritos  Configuração  Ajuda
~: julia
(base) bamorim@bamorim-AMD-4750g:~$ julia
Documentation: https://docs.julialang.org
Type "?" for help, "]" for Pkg help.
Version 1.5.3 (2020-11-09)
Official https://julialang.org/ release

julia> 1+1
2

julia> 
```

```
JupyterLab
localhost:8888/lab
File Edit View Run Kernel Tabs Settings Help
Untitled.ipynb
Julia 1.5.3
[1]: versioninfo()
Julia Version 1.5.3
Commit 788b2c77c1 (2020-11-09 13:37 UTC)
Platform Info:
  OS: Linux (x86_64-pc-linux-gnu)
  CPU: AMD Ryzen 7 PRO 4750G with Radeon Graphics
  WORD_SIZE: 64
  LIBM: libopenlibm
  LLVM: libLLVM-9.0.1 (ORCJIT, znver2)

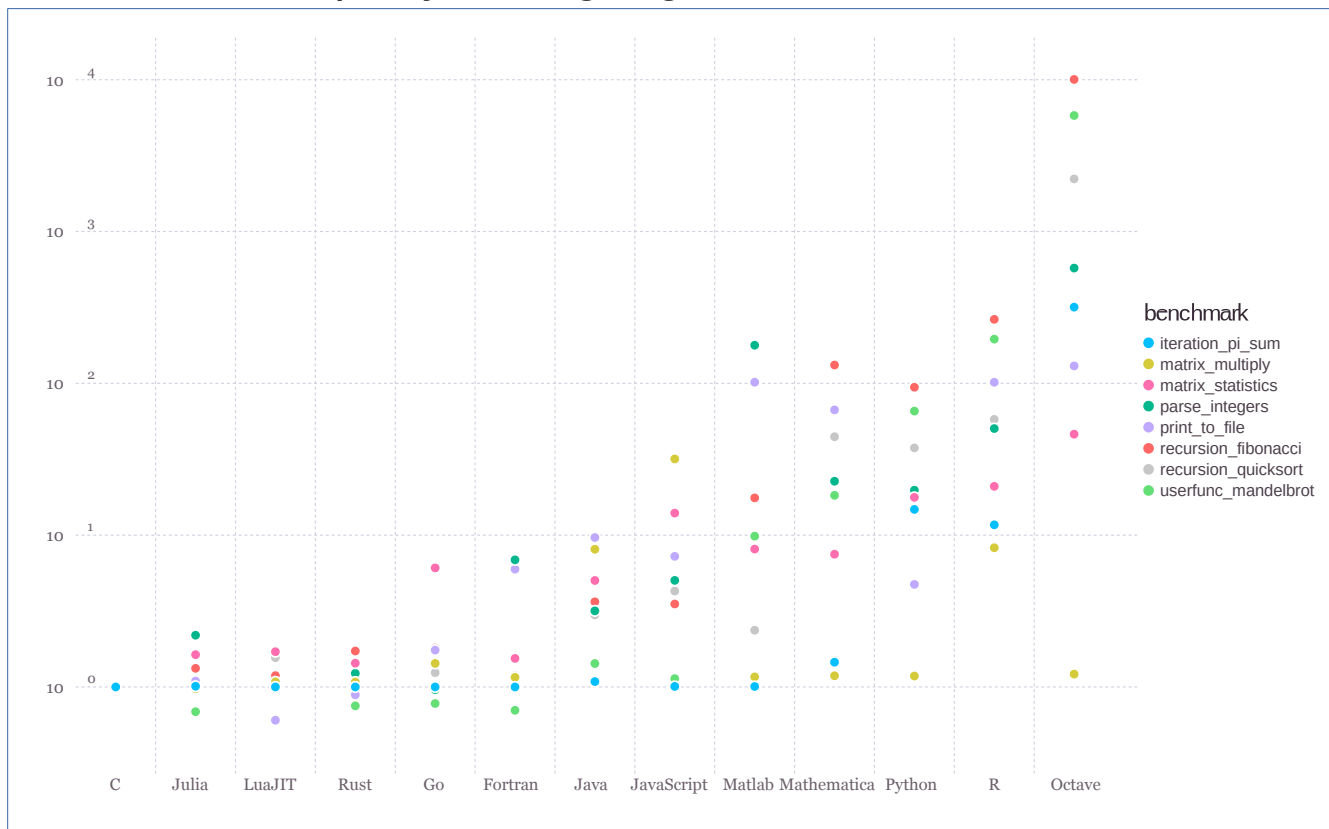
[2]: 1+1
[2]: 2

[ ]: 
```

Julia is a high performance language

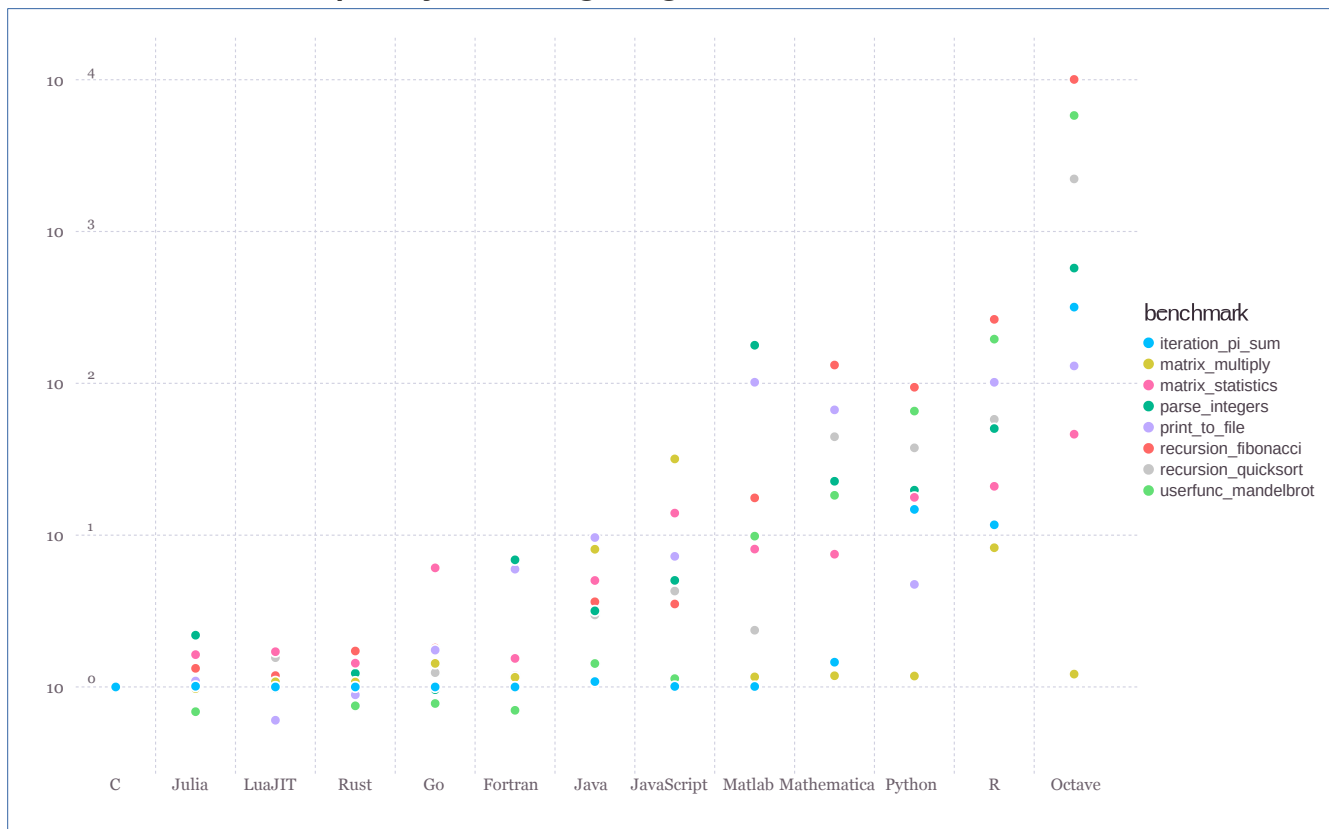
Julia is a high performance language

Julia microbenchmarks from <https://julialang.org/benchmarks/>



Julia is a high performance language

Julia microbenchmarks from <https://julialang.org/benchmarks/>



Julia is actually a compiled language!
It uses LLVM (same backend as the Clang C compiler)

Julia has nice syntax

Julia has nice syntax

Let us define a function that returns a rotation by an angle that is a multiple of $2\pi/6$:

Julia has nice syntax

Let us define a function that returns a rotation by an angle that is a multiple of $2\pi/6$:

Python code:

```
import numpy as np

def c6rotation(n):

    return np.array([
        [np.cos(2*np.pi*n/6), -np.sin(2*np.pi*n/6)],
        [np.sin(2*np.pi*n/6), np.cos(2*np.pi*n/6)]
    ])
```

Julia has nice syntax

Let us define a function that returns a rotation by an angle that is a multiple of $2\pi/6$:

Julia code:

```
c6rotation(n) = [  
    cos(2π*n/6) -sin(2π*n/6);  
    sin(2π*n/6) cos(2π*n/6)  
]
```

Python code:

```
import numpy as np  
  
def c6rotation(n):  
    return np.array([  
        [np.cos(2*np.pi*n/6), -np.sin(2*np.pi*n/6)],  
        [np.sin(2*np.pi*n/6), np.cos(2*np.pi*n/6)]  
    ])
```

Maybe this sparked your interest

But you might be asking:

“Is anyone using Julia?”

Julia adoption

Julia adoption

Julia adoption evolution between 1st Jan 2020 and 1st Jan 2021:

	Total Cumulative as of Jan 1, 2020	Total Cumulative as of Jan 1, 2021	Change
Number of Julia Downloads (JuliaLang + Docker + JuliaPro)	12,950,630	24,205,141	+87%
Number of Julia Packages	2,787	4,809	+73%
GitHub stars (Julia language repo + registered packages)	99,830	161,774	+62%
YouTube views (Julia language channel)	1,562,223	3,320,915	+113%
Published citations of Julia: A Fast Dynamic Language for Technical Computing (2012) + Julia: A Fresh Approach to Numerical Computing (2017)	1,680	2,531	+51%
Discourse posts	137,399	211,888	+54%
TIOBE Index Rank	#47	#23	+24

from: <https://juliacomputing.com/blog/2021/01/newsletter-january/>

Julia adoption

Julia adoption evolution between 1st Jan 2020 and 1st Jan 2021:

	Total Cumulative as of Jan 1, 2020	Total Cumulative as of Jan 1, 2021	Change
<u>Number of Julia Downloads</u> (JuliaLang + Docker + JuliaPro)	12,950,630	24,205,141	<u>+87%</u>
<u>Number of Julia Packages</u>	2,787	4,809	<u>+73%</u>
GitHub stars (Julia language repo + registered packages)	99,830	161,774	+62%
YouTube views (Julia language channel)	1,562,223	3,320,915	+113%
Published citations of Julia: A Fast Dynamic Language for Technical Computing (2012) + Julia: A Fresh Approach to Numerical Computing (2017)	1,680	2,531	+51%
Discourse posts	137,399	211,888	+54%
TIOBE Index Rank	#47	#23	+24

from: <https://juliacomputing.com/blog/2021/01/newsletter-january/>

Companies using Julia

Companies using Julia

From <https://juliacomputing.com>:

JULIA USERS AND JULIA COMPUTING CUSTOMERS



Companies using Julia

From <https://juliacomputing.com>:

JULIA USERS AND JULIA COMPUTING CUSTOMERS



Science Labs

LINCOLN LABORATORY
Massachusetts Institute of Technology

OAK
RIDGE
National Laboratory



Companies using Julia

From <https://juliacomputing.com>:

JULIA USERS AND JULIA COMPUTING CUSTOMERS

Tech



Companies using Julia

From <https://juliacomputing.com>:

JULIA USERS AND JULIA COMPUTING CUSTOMERS

Finance/Banking/Assurance



Companies using Julia

From <https://juliacomputing.com>:

JULIA USERS AND JULIA COMPUTING CUSTOMERS

Pharma



Companies using Julia

From <https://juliacomputing.com>:

JULIA USERS AND JULIA COMPUTING CUSTOMERS

Covid?



Most importantly:

Julia is being taught at universities!

Julia course at MIT

Introduction to your professors | Week 1 | 18.S191 MIT Fall 2020

42 572 visualizações • 01/09/2020

1,2 MIL 7 PARTILHAR GUARDAR

The Julia Programming Language
54 mil subscritores

SUBSCRITO

For full course information, visit <https://github.com/mitmath/18S191>
Lecture 1: <https://youtu.be/DGgJ9xcCfG>

MOstrar Mais

Computational thinking | MIT 18.S191 Fall 2020
The Julia Programming Language - 1/59

- 1 Introduction to your professors | Week 1 | 18.S191 MIT Fall 2020
The Julia Programming Language
- 2 Working with images in Julia | Week 1 | 18.S191 MIT Fall 2020 | Grant...
The Julia Programming Language
- 3 How to install Julia and Pluto | Week 1 | 18.S191 MIT Fall 2020
The Julia Programming Language
- 4 First taste of abstraction with arrays | Week 1 | 18.S191 MIT Fall 2020
The Julia Programming Language
- 5 A programming language to heal the planet together: Julia | Alan Edelman ...
TEDx Talks
- 6 Basics of arrays in Julia | Week 1 | 18.S191 MIT Fall 2020
The Julia Programming Language

Todos Programação de computadores Relações

Convolutions
Computational Thinking 36:10

Convolutions in image processing | Week 1 | MIT...
The Julia Programming Language
263 mil visualizações • Transmido há 6 meses

Course Welcome + Intro to Arrays & Images! MIT...
The Julia Programming Language
6,8 mil visualizações • Transmido há 1 mês

[illegible]

Julia course at MIT

It is being taught for some time:

- Massachusetts Institute of Technology (MIT)
 - [6.251 / 15.081](#), Introduction to Mathematical Programming (Prof. Dimitris J. Bertsimas), Fall 2015
 - [18.06](#), Linear Algebra: Fall 2015, Dr. [Alex Townsend](#); Fall 2014, Prof. Alexander Postnikov; Fall [2013](#), Prof. Alan Edelman
 - [18.303](#), Linear Partial Differential Equations: Analysis and Numerics (Prof. [Steven G. Johnson](#)), Fall 2013–2016.
 - [18.337 / 6.338](#), Numerical Computing with Julia (Prof. [Alan Edelman](#)). [Fall 2015 \(IJulia notebooks\)](#). Fall 2013–
 - [18.085 / 0851](#), Computational Science And Engineering I (Prof. Pedro J. Sáenz)
 - [18.330](#), Introduction to Numerical Analysis (Dr. Homer Reid), Spring 2013–2015
 - [18.335](#), Introduction to Numerical Methods (Prof. Steven G. Johnson), Fall 2013, Spring 2015
 - [18.338](#), Eigenvalues Of Random Matrices (Prof. Alan Edelman), Spring 2015
 - [18.S096](#), Performance Computing in a High Level Language (Steven G. Johnson, Alan Edelman, David Sanders, Jeff Bezanson), January 2017.
 - [15.093 / 6.255](#), Optimization Methods (Prof. Dimitris Bertsimas and Dr. Phebe Vayanos), Fall 2014
 - [15.S60](#), Software Tools for Operations Research (Iain Dunning), Spring 2014
 - [15.083](#), Integer Programming and Combinatorial Optimization (Prof. Juan Pablo Vielma), Spring 2014

From: <https://julialang.org/learning/classes/>

Julia in other universities:

From: <https://julialang.org/learning/>



For a more detailed list: <https://julialang.org/learning/classes/>

Projects using Julia: Celeste project

Projects using Julia: Celeste project

Project to catalog Sloan Digital Sky Survey

- processed ~ 178 terabytes of data in 14.6 minutes
- peak performance: 1.54 **petaflops**
- used **1.3 million threads** running on **9 300 nodes** at NERSC
(National Energy Research Scientific Computing Center)
- written **entirely in Julia**

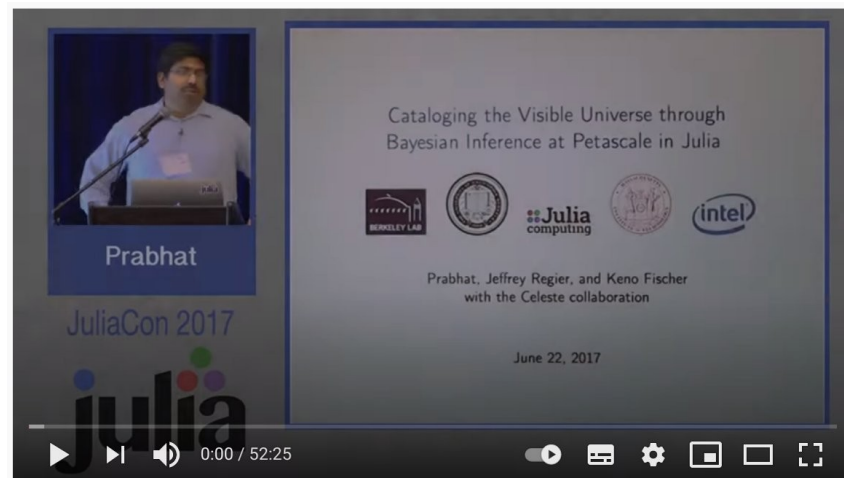
Projects using Julia: Celeste project

Project to catalog Sloan Digital Sky Survey

- processed ~ 178 terabytes of data in 14.6 minutes
- peak performance: 1.54 **petaflops**
- used **1.3 million threads** running on **9 300 nodes** at NERSC
(National Energy Research Scientific Computing Center)
- written **entirely in Julia**

For more info:

- <https://juliacomputing.com/case-studies/celeste/>
- <https://youtu.be/uecdcADM3hY>
- <https://github.com/jeff-regier/Celeste.jl>



JuliaCon 2017 | Celeste.jl: Petascale Computing in Julia | Prabhat, Regier & Fischer

6182 visualizações • 05/08/2017

61

1

PARTILHAR

GUARDAR

...

Projects using Julia: Clima project

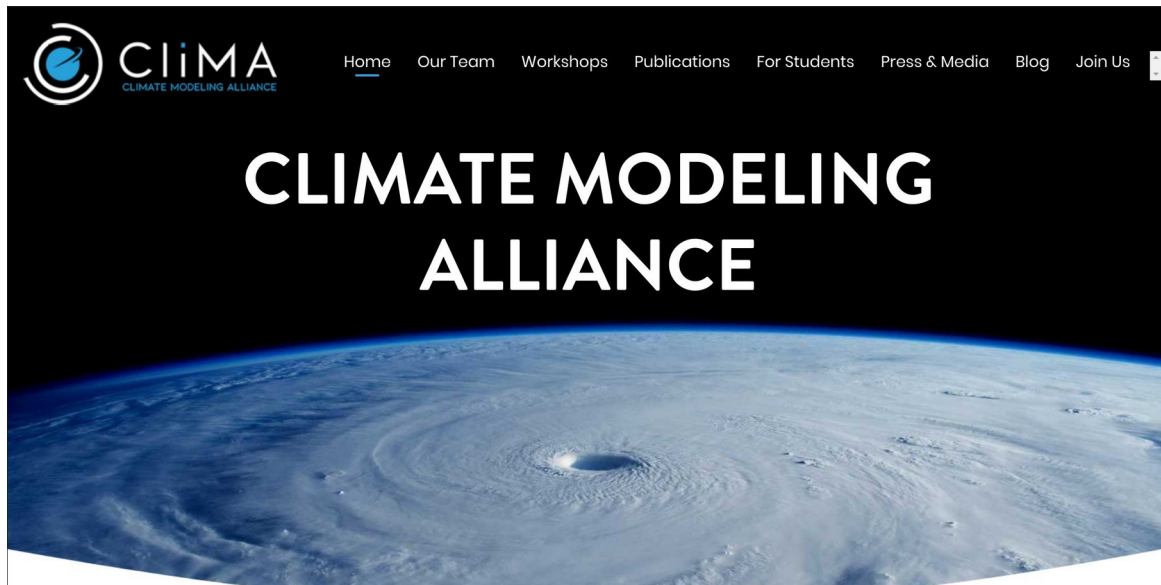
Projects using Julia: Clima project

New climate model built from the ground up in Julia

<https://clima.caltech.edu>

Research consortium:

- Caltech
- MIT
- Naval Postgraduate School
- Jet Propulsion Laboratory



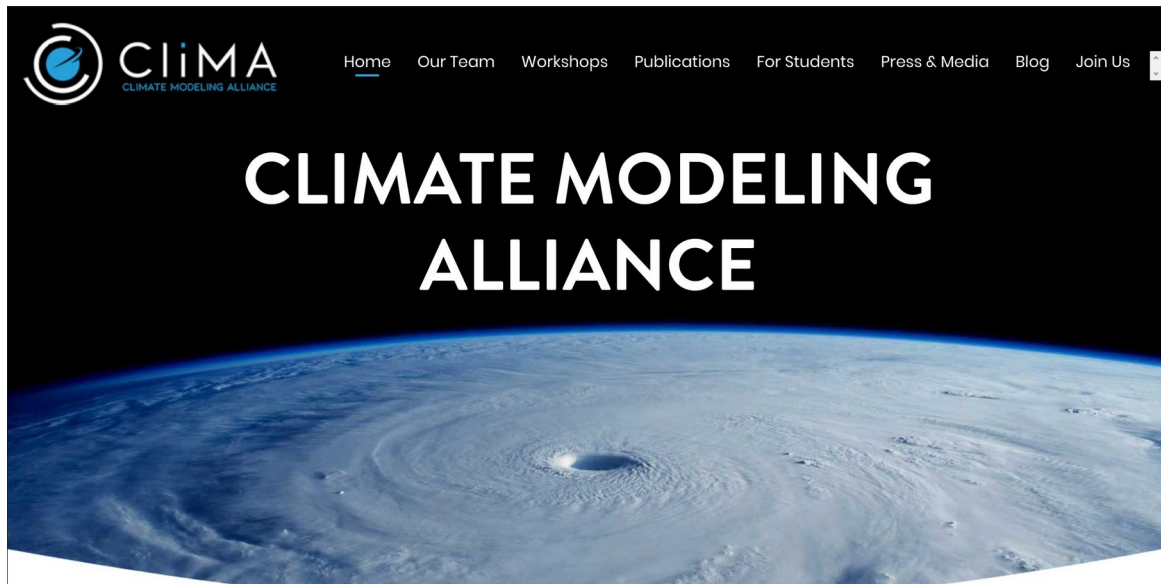
Projects using Julia: Clima project

New climate model built from the ground up in Julia

<https://clima.caltech.edu>

Research consortium:

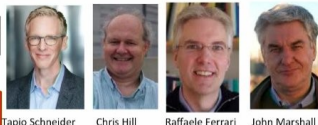
- Caltech
- MIT
- Naval Postgraduate School
- Jet Propulsion Laboratory



MIT News: New climate model to be built from the ground up

December 12, 2018

To take advantage of new computer architectures, languages, and machine learning techniques, the team has partnered with [Alan Edelman](#)'s group in CSAIL at MIT, who will help write the new generation climate model in the Julia computing language developed by the group. This will enable the MIT team to target GPUs, CPUs, and evolving computer architectures within one code base. **get the factor of three I'm happy to report it was three times faster and**



Tapio Schneider Chris Hill Raffaele Ferrari John Marshall

Where OK with a 3x slowdown.
Ended up being **3x faster!**

<https://youtu.be/nwdGsz4rc3Q?t=1238>

<https://www.hpcwire.com/2020/01/14/julia-programmings-dramatic-rise-in-hpc-and-elsewhere/>

Rich package ecosystem

Rich package ecosystem

- **Iterative Linear Solvers:** IterativeSolvers.jl, KrylovKit.jl
- **Eigensolvers:** ArnoldiMethod.jl, KrylovKit.jl
- **Differential equations:** DifferentialEquations.jl
- **Integration:** QuadGK.jl, FastGaussQuadrature.jl, Quadrature.jl
- **Optimization:** Optim.jl, JuMP.jl
- **Plotting:** Plots.jl, Pyplot.jl, VegaLite.jl, Makie.jl, Gadfly.jl
- **Machine learning:** Flux.jl, Knet.jl

Rich package ecosystem

- **Iterative Linear Solvers:** IterativeSolvers.jl, KrylovKit.jl
- **Eigensolvers:** ArnoldiMethod.jl, KrylovKit.jl
- **Differential equations:** DifferentialEquations.jl
- **Integration:** QuadGK.jl, FastGaussQuadrature.jl, Quadrature.jl
- **Optimization:** Optim.jl, JuMP.jl
- **Plotting:** Plots.jl, Pyplot.jl, VegaLite.jl, Makie.jl, Gadfly.jl
- **Machine learning:** Flux.jl, Knet.jl

Physics related packages:

- **Tensor Networks:** Itensors.jl
- **Quantum Computing:** Yao.jl
- **Quantum optics:** QuantumOptics.jl
- **DFT:** DFTK.jl
- **Quantum Monte Carlo:** MonteCarlo.jl
- **Tight-binding:** Quantic.jl

Rich package ecosystem

- **Iterative Linear Solvers:** IterativeSolvers.jl, KrylovKit.jl
- **Eigensolvers:** ArnoldiMethod.jl, KrylovKit.jl
- **Differential equations:** DifferentialEquations.jl
- **Integration:** QuadGK.jl, FastGaussQuadrature.jl, Quadrature.jl
- **Optimization:** Optim.jl, JuMP.jl
- **Plotting:** Plots.jl, Pyplot.jl, VegaLite.jl, Makie.jl, Gadfly.jl
- **Machine learning:** Flux.jl, Knet.jl

Physics related packages:

- **Tensor Networks:** Itensors.jl
- **Quantum Computing:** Yao.jl
- **Quantum optics:** QuantumOptics.jl
- **DFT:** DFTK.jl
- **Quantum Monte Carlo:** MonteCarlo.jl
- **Tight-binding:** Quantica.jl

You can search for more using:

- juliahub.com/ui/Packages
- juliaobserver.com/packages
- just use google

More resources to learn about julia

- **Julia page:** julialang.org
- **Julia documentation:** docs.julialang.org
- **Julia forum:** discourse.julialang.org
- **Julia youtube channel:** www.youtube.com/user/JuliaLanguage/playlists

More tutorials and books are listed in julialang.org/learning