[11pt,a4paper,english,greek,twoside]thesis graphicx epstopdf indentfirst verbatim amsmath amsthm amssymb latexsym hellas hyphenat makeidx algpseudocode algorithm [hyphens]url enumitem xspace booktabs multirow subfig tabularx listings xcolor bbding footmisc belowskip=10pt,aboveskip=15pt

1.2

[1]1 [1]1

[1]english1

[1]1

92

codegreenrgb0,0.6,0 codegrayrgb0.5,0.5,0.5 codepurplergb0.58,0,0.82 backcolourrgb0.95,0.95,0.92

mystyle backgroundcolor=backcolour, commentstyle=codegreen, keywordstyle=magenta, numberstyle=codegray, stringstyle=codepurple, basicstyle=, breakatwhitespace=false, breaklines=true, captionpos=b, keepspaces=true, numbers=left, numbersep=5pt, showspaces=false, showstringspaces=false, tabsize=2

blue red json backgroundcolor=backcolour, showstringspaces = false, keywords = false,true, alsoletter = 0123456789., morestring = [s]"", stringstyle = , MoreSelectCharTable =@DefSaveDef‘::@json@json, basicstyle = , keywordstyle = , @json:@json@mode=@Pmode

@AddToHookOutput@mode=@Pmode@DetectKeywords

@AddToHookEOL

proposition theorem corollary lemma example remark notation law chapter.definition chapter.proposition chapter.theorem chapter.corollary chapter.lemma chapter.example [1]{1} $\Longrightarrow$

greek

document greek

Julia: , ux3c4ux3bf-ux3c0ux3b5ux3c1ux3b9ux3b2ux3acux3bbux3bbux3bfux3bd-julia-ux3bcux3b5ux3bbux3adux3c ux3baux3b1ux3b9-ux3b5ux3c6ux3b1ux3c1ux3bcux3bfux3b3ux3adux3c2

Abstractabstract

ellinika?

Description of the Julia programming languagedescription-of-the-julia-programming-language

Julia is a high level, multi-paradigm, dynamically typed, programming language. It is aimed at fields such as numerical analysis, computational science, while also being well suited for general purpose programming.

Focusing on providing Ruby's dynamic types, the syntactic simplicity of Python and C-like performance, Julia could prove to be the leading language in high performance computing, and an indispensable tool for research in scientific and engineering fields.

Julia is released under the MIT license, therefore is free and open source.

Feature overviewfeature-overview

Julia is JIT-compiled and garbage collected and uses multiple dispatch. It is designed with high performance in mind, being comparable to much lower level languages, such as C. In addition, parallel execution and distributed computing are first class citizens. Other key components, include macros and metaprogramming support, a built in package manager, seamless interop with C and Fortran and a highly sophisticated compiler, able to generate specialized code, depending on argument types.

Historyhistory

Julia was designed by Viral B. Shah, Jeff Bezanson, Stefan Karpinski and Alan Edelman. Released in 2015, after first being revealed on Valentine's Day of 2012. Its user base has been growing exponentially, while its popularity landed it at the top 50 of the TIOBE Programming Community Index (www.tiobe.com).

The two languages problemthe-two-languages-problem

Julia's design, came as an answer to the two languages problem, faced by modern data scientists; Writing a code prototype in a dynamically typed language, to verify a working solution, but then having to rewrite a whole new implementation in another, statically typed language in order to achieve acceptable performance.

One can easily implement some algorithm or conceptual solution in Julia. Its great advantage in comparison to languages like Python, is that the very same code, can achieve the performance of highly optimized, machine specific code (thanks to LLVM), only by introducing very minor changes, in the form of type decla-

ration for a method's arguments. This makes the code extremely easy to optimize, even for users with little understanding of low level architecture.

This is achieved, thanks to Julia's versatile and advanced compiler, that can produce LLVM IR (intermediate representation), specialized on the types of the parameters of each calculation. If the types are not known in advance, the generated assembly may not make any assumptions about the arguments' memory representation, and while perfectly working, it is sub optimal. In case that constraints are enforced on the types, the compiler is smart enough, to take advantage of them, and generate assembly similar to that of the statically typed C. As a result, you have all of the benefits of a statically typed language, both in type safety and performance, as an opt in feature, allowing the liberty and ease of use of a dynamic type system wherever speed is not a concern.

Platformsplatforms

Julia is JIT-compiled with an LLVM backend. It can generate native code for all of the major modern platforms:

itemize
W indows
L inux
M ac OS
F reeBSD