EGU24-16244, updated on 18 May 2024
https://doi.org/10.5194/egusphere-egu24-16244
EGU General Assembly 2024

Building an ecosystem of computational geosciences software: why Julia?

Albert de Montserrat Navarro[1], Ludovic Räss[2], Boris Kaus[3], Ivan Utkin[1], and Pascal Aellig[3]
[1]ETH Zurich, Switzerland
[2]University of Laussane, Switzerland
[3]Johannes Gutenberg-Universität Mainz, Germany

Traditionally, the Earth science community has relied on statically compiled and explicitly typed, long-lived programming languages, namely C/C++ and Fortran, for computationally intensive production runs. In contrast, dynamic languages such as Python and MATLAB have served as alternatives for less computationally demanding tasks, prototyping, visualisation and teaching. The lower entry-level of the latter is often used as a "glue language" where the performance-critical code is written in a static language. In the recent decades other modern languages (C#, Nim, Go, Rust...) have been developed, but so far none of them have had a significant impact on the general scientific community. Among the new modern programming languages there is Julia -with the 1.0 version being released just in 2018- which was designed with the computational scientific community as the main target user base.
Julia's main appeal for computational science are (i) performance, it produces compiled machine code with performance potentially similar to other statically compiled languages; (ii) interactivity, it is a dynamic language, and most of the development and prototyping can be done in interactive sessions using its built-in REPL (read-eval-print loop); and (iii) readability, code is *often* easier to read than e.g. C++ or Rust. Julia also natively supports linear algebra operations (e.g. it ships its own wrappers for libraries such as OpenBlas, SuiteSparse or LAPACK), multi-threading and distributed parallelism, which are crucial for a vast number of scientific applications.

Additionally, Julia offers a series of features that are not intrinsic to the language itself which can ease and improve the user experience compared to traditional C/C++/Fortran: (i) Julia has a well-built package manager where one can easily add and install any version of any registered third-party package by typing two words in the REPL; (ii) Julia itself is open source, as well as all the registered packages (hosted in Git repositories), and discourages black-box software; (iii) reproducibility is easy thanks to .toml files containing what exact version packages are needed to be installed.

Here we will discuss how the combination of the previous points, along with other important features of the language (multiple dispatch, metaprogramming,...), and existing scientific libraries (GPU support, auto-differentiation,...) make Julia an excellent platform for the geoscientific community to build an ecosystem of open-source and highly modular software, opposite to the

classic lone-wolf large monolithic code. Finally, we will present an overview of the current status of the JuliaGeodynamics and related organisations, as well as the general Earth sciences ecosystem.