



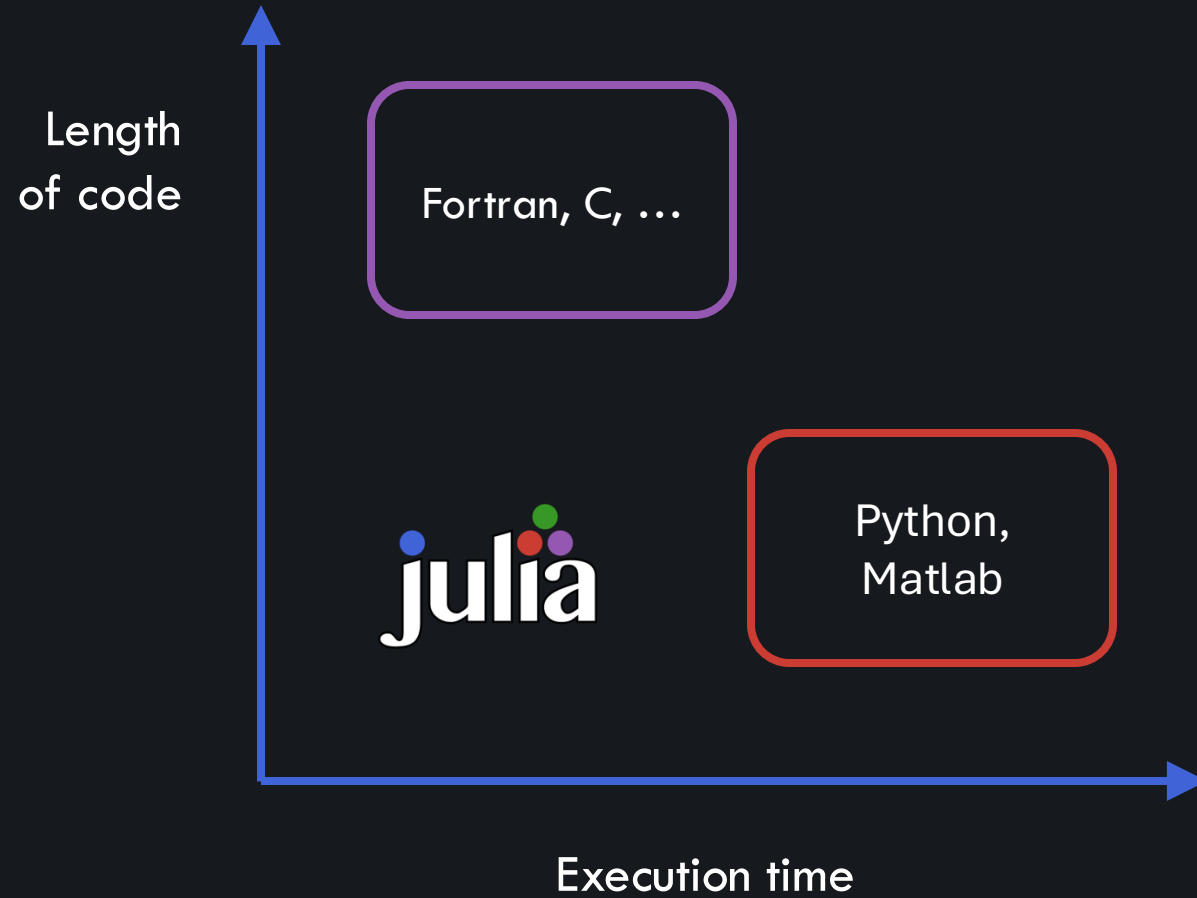
Utrecht University

Programming **in** **julia**
Tools for dynamical systems
and climate

Reyk Börner

Python for Lunch Seminar, 18 September 2025

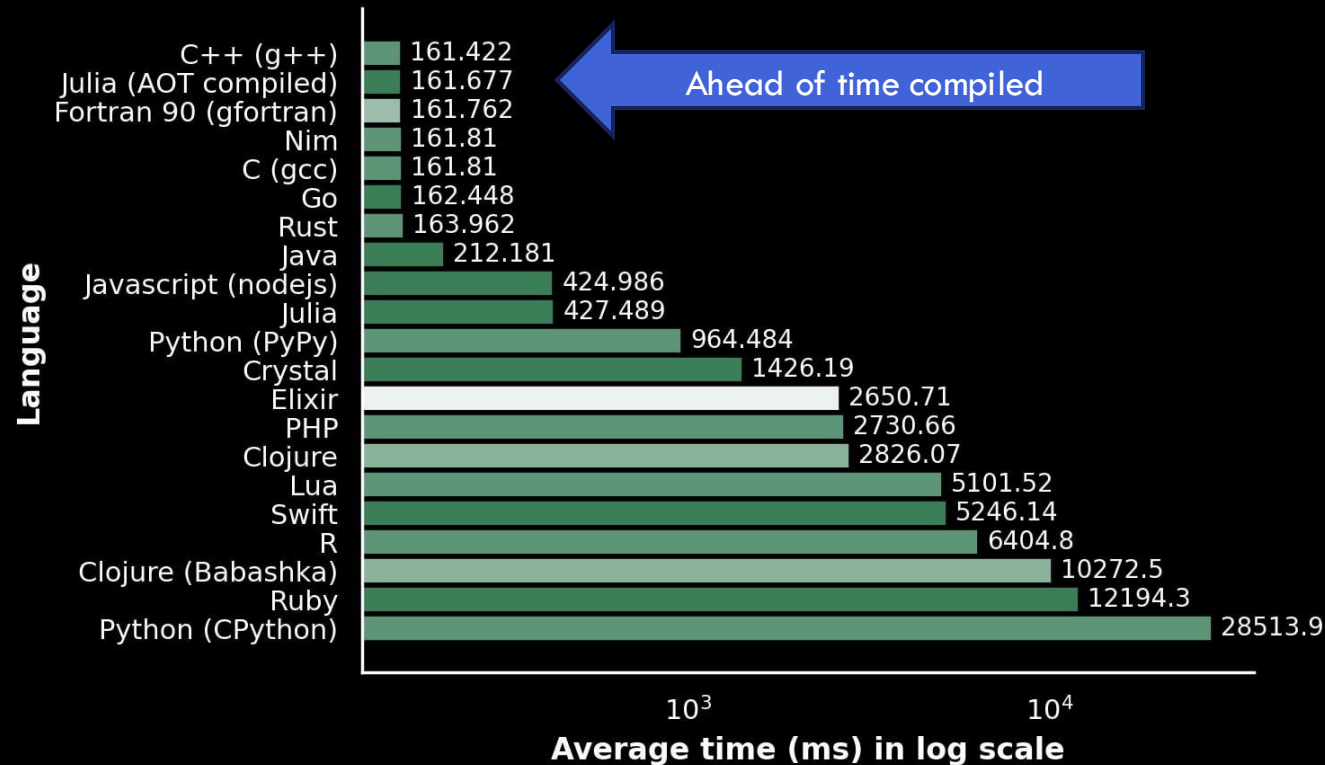
Two-language problem



Two-language problem

Speed comparison of various programming languages

Method: calculating π through the Leibniz formula 100000000 times



Generated: 2022-10-16 19:55

<https://github.com/niklas-heer/speed-comparison>

Two-culture problem



The diagram consists of two rounded rectangular boxes side-by-side. The left box is blue and contains the text 'Climate scientist'. The right box is purple and contains the text 'Software engineer'. Below these boxes is a quote.

Climate scientist

Software engineer

“A typical Julia user is already 90% of the way to a software developer”

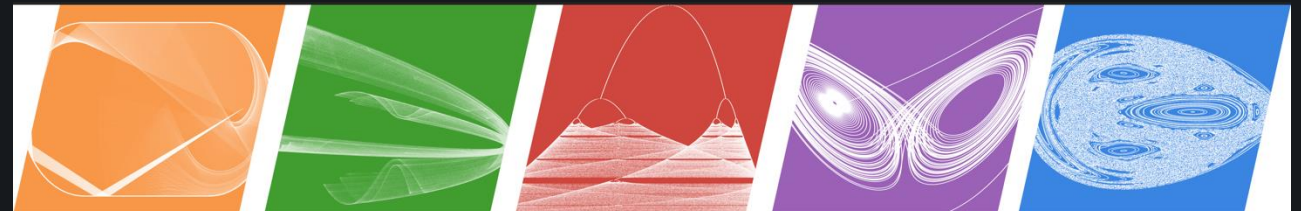
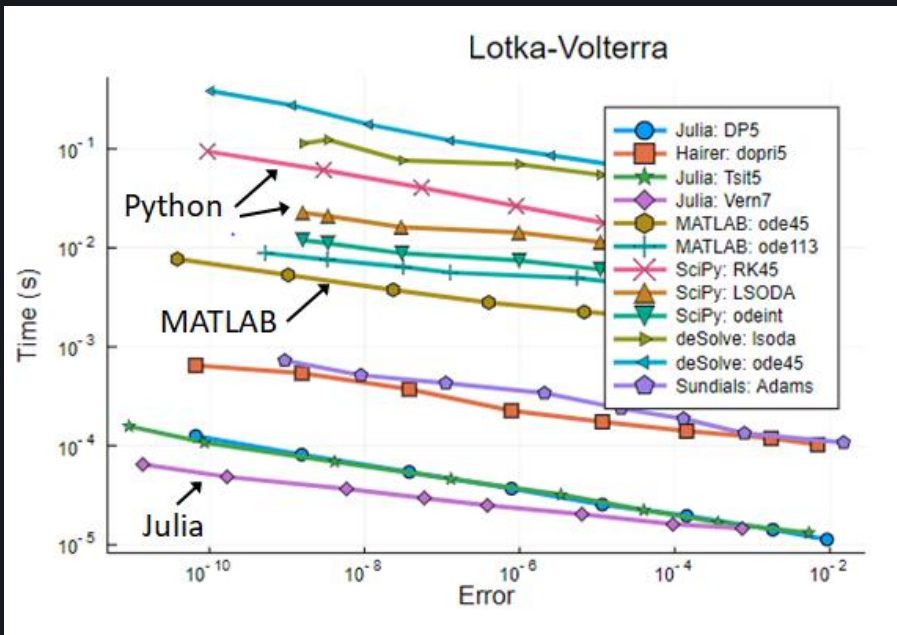
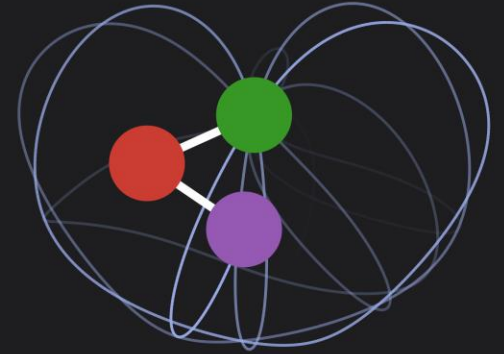
What should a good scientific coding language offer?

- Fast to write
- Fast to run
- Composable, clearly structured (easy to read)
- Extendible
- Accessible
- Reproducible

Julia for dynamical systems

- SciML
- DifferentialEquations.jl
- JuliaDynamics
- DynamicalSystems.jl
- BifurcationKit.jl
- CriticalTransitions.jl

Dynamical
Systems.jl



JuliaDynamics

open source software for nonlinear dynamics and chaos

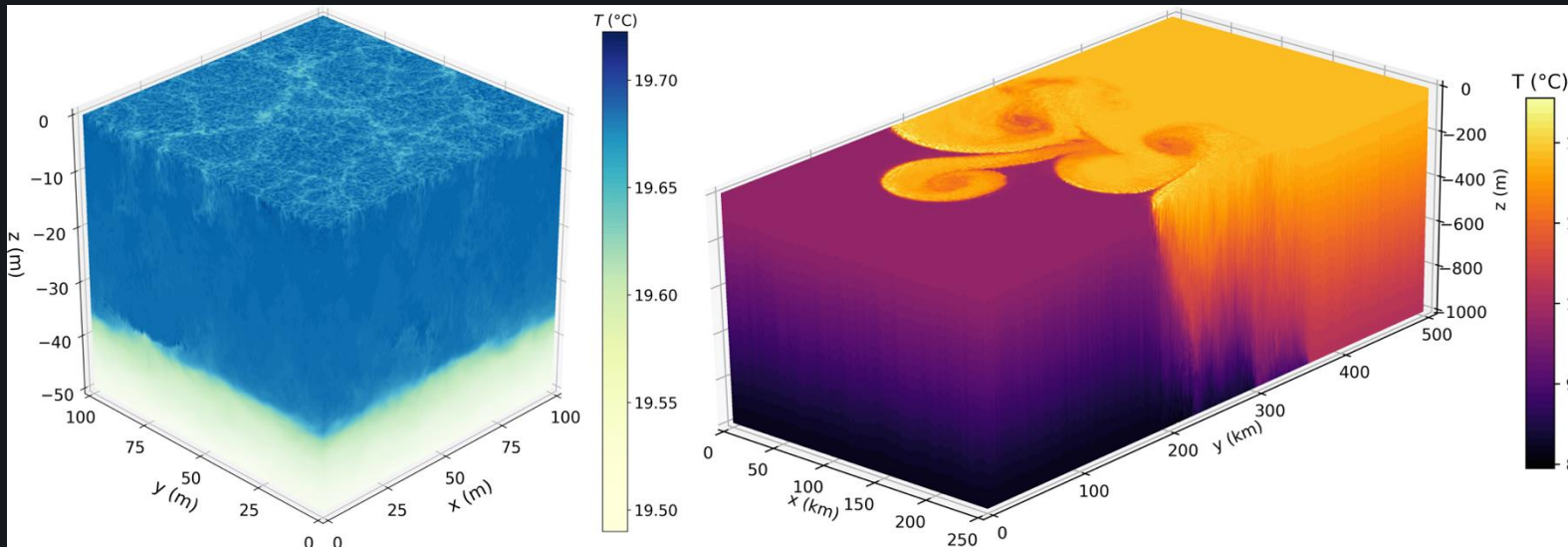
Julia for climate modeling

- CliMA
 - ClimaOcean.jl
 - ClimaAtmos.jl
 - ClimaLand.jl
 - ...
- SpeedyWeather.jl
- Oceananigans.jl



Oceananigans.jl

 Fast and friendly fluid dynamics on CPUs and GPUs.



SpeedyWeather.jl

Writing your own Julia package

... see example: `DoubleTrouble.jl`

Contributing to DynamicalSystems.jl


CoupledODEs

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}, p, t)$$

CoupledSDEs

$$d\mathbf{u} = \mathbf{f}(\mathbf{u}, p, t)dt + \mathbf{g}(\mathbf{u}, p, t)d\mathcal{N}_t$$

CriticalTransitions.jl



CriticalTransitions.jl

Search docs (Ctrl + /)

Home

- Purpose

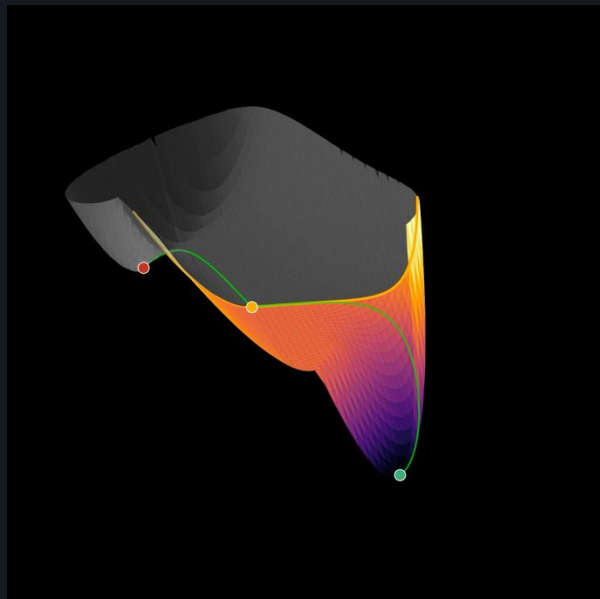
Home

[GitHub](#) [🔗](#) [⚙️](#) [^](#)

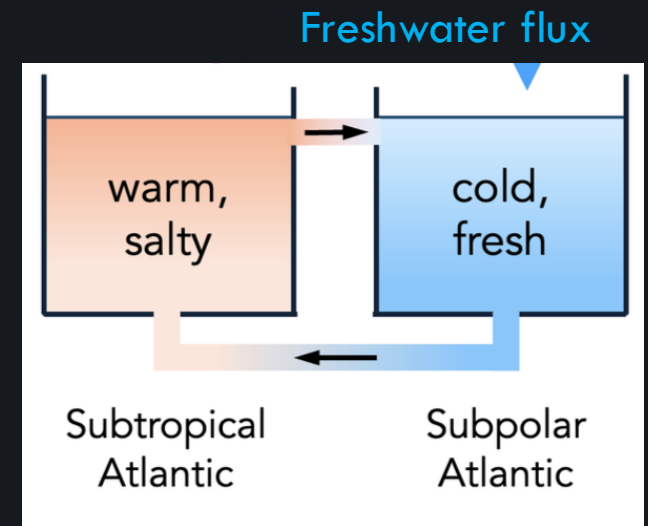
CriticalTransitions.jl

A Julia package for numerical investigation of noise- and rate-induced transitions in dynamical systems.

This package provides a toolbox for dynamical systems under time-dependent forcing, with a focus on tipping phenomena and metastability. Building on [DynamicalSystems.jl](#) and [DifferentialEquations.jl](#), the code adds functionality specifically to study stochastic and non-autonomous dynamics – while keeping a familiar user interface and taking advantage of the powerful existing solvers in Julia.



Example:
Stommel box model
(Stommel 1961)



Resources

- Julia homepage: <https://julialang.org>
- Juliaup (installer): <https://github.com/JuliaLang/juliaup>
- Zero2Hero tutorial: <https://github.com/Datseris/Zero2Hero-JuliaWorkshop>
- Good scientific code workshop: <https://github.com/JuliaDynamics/GoodScientificCodeWorkshop>

Thanks to

Orjan Ameye,
George Datseris,
Ryan Deeley,
Frank Hellmann,
Milan Klöwer,
Raphael Römer,
Paul Schultz